Hybrid Deep Learning Long Tail Services Recommendation System:An approach to solve the problem of long tail services .

Er.Meenakshi Sharma, Dr.Satpal

[1*]Computer Science Department, Baba Mastnath University, Haryana , India

2 Computer science Department ,BabaMastnath University,Haryana ,India

Abstract - There are number of online services and size of users have increased dramatically over the past years. In long-term web services are important to increase the web API economy, how to recommend the long-tail web services efficiently is a primary issue. Moreover, to focus on this problem, the traditional web based recommendation services performs poorly on long-tail side. To overcome the problem of severe sparsity of historical usage data and unsatisfactory quality of description content, we are focusing on Convolutional Neural Network approach to boost the performance of the network and reduce time consumption. Recommendation engine collects and saves suggestions from individuals who know about the decisions that they confronted and furthermore it values their points of view and identifies them as the specialists. Two ordinary entities which seem in any recommendation engine are the products and individuals. A DNN involved convolutional layer exchange with maxpooling layer pursued by completely associated layers and a last classification layer .

Keywords :CNN, Recommendation system, Long -Tail. Sparse Data.

I.INTRODUCTION :

Recommender Systems based on Long tail keywords:

The idea has discovered some basic for experiments, investigation and application. The term utilized in E-business,user-driven, microfinance, mass-media novelty and social-network instruments financial models, promoting a frequency-distribution with a long-tail has been considered by analysts since in any event year 1946.The term has likewise been utilized in the insurance&finance business for a long time.

In recommender system, “long tail” items are considered to be particularly valuable. Many clustering algorithms based on CF designed only to tackle the “long tail” items, while others trade off the overall recommendation accuracy and “long tail” performance.

Our approach is based on utilizing the item popularity information. We demonstrate that “long tail” recommendation can be inferred precisely by balanced item popularity of each cluster.

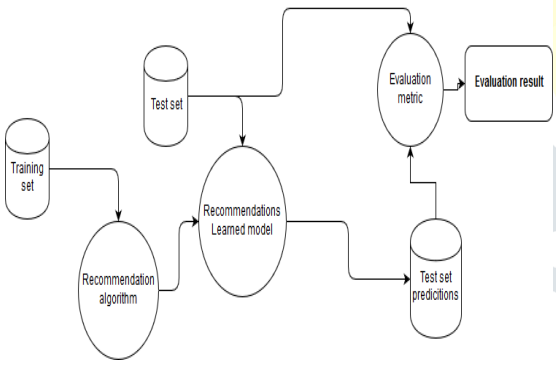
1.1 Functions of Recommendation system:

A recommendation engine is application or system that encourages the client to choose an appropriate thing or finding pertinent data amongst a lot of candidates utilizing a knowledge-base that can any of be hand coded by specialists or gained from practices of the clients. Regularly, a recommender makes 3 of functions:

- **Information Collection:** The system gathers all the usable data for the forecast undertaking including the clients' properties, practices, or the content of the assets the client gets to.
- **Learning:** It applies a learning algo for filtering and exploiting the features of clients from the gathered data.
- **Prediction:** It infers the sort of resources the consumer might incline toward are at that point made either legitimately dependent on the dataset gathered in the phase of data collection (memory-based predictions) or with a model gained from it (model-based predictions).

1.2 Work Flow Of Recommendation System:

Frequently called as Recommendation Engines, they are basic algos which objective to give the most important exact things to the client by separating valuable stuff from of a immense pool of data -base. Recommender finds data-patterns in the dataset by learning clients picks and gives the results that co-identifies with their interests &needs.



II. Problem in Existing Data:

As the demand of web services and API by developers is increasing day to day to build the massive applications and fetch the high amount of data. The long-tail web services works efficiently for recommendation methods. The main issue we face for long-tail web services is data sparsity. Various other methods also applied to remove this problem, for this research, we used Stacked Denoising Auto encoders (SDAE) which is widely discussed for data sparsity. Its basic idea is to learn the patterns of developers' preference instead of modeling individual services. However in past [14] only the historical usage data was assumed as sparsity. This is the issue of considerations, in our research we will be going to give stress on more parameters. To overcome this issue and

achieving the maximum accuracy, we have finalized three objectives, which propose the hybrid deep learning techniques using Python libraries.

However in past [14] only the historical usage data was assumed as sparsity. This is the issue of considerations, in our research we will be going to give stress on more parameters such as: QoS, user profiles and social connection between services. Moreover to boost the output performance we will be going to investigate more sophisticated deep learning models such as convolutional neural networks (CNN) for Classification.

III. How to use Long-tail Services

Long tails are less usable services, so in this research, a services which has been used maximum 5 times to be considered a long-tail service. On the other side, some hot services are taking too much popularity which are equal or more than 5 times usage.

Netflix is maximum attributed to a "long tail" Services. Therefore the maximum of their inventory is not in higher demand, these products, unavailable at limited competitors,create a fix fraction of total income in aggregate. Additionally, long- tail product available boosts their head sales by offers consumers the convenience of "one-stop shopping" for niche tastes and their mainstream. However, many recommendation systems, build using collaborative filtering methods, can't recommends the long-tail products due to issue of data sparsity. It largely acknowledge the recommended famous products which is easy for more trivial while to recommending long tail products addition to more new yet it is also a lot of challenging task.

1) **Challenges:-**

a) Sparsity of the data : lack of information in result is called sparsity in this only few item are rated by the user.

- Though, it is problematic to differentiate the comparable interests amongusersdue to the issue of sparsity is caused by the inadequate no. of the dealings and feedback info, which restrained the use of the CF

.

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

Fig : 2.7Sparse data

- Data Sparsity has a negative effect on the excellence of recommendations that can be provided by traditional Collaborative Filtering algorithms.
- Because of the sparsity level in the data-sets, usually deal with a user can seem similarly similar to any additional if the comparison metrics utilized are not delicate adequate.
- For itself, the individual is simply offered with suggestions for the most prevalent product with a cluster of 'randomly' chosenindividuals; while there can be a possibly higherextent of divergence amongst associates of this cluster.

b**) Cold –start problem**: a condition where a recommend does not have right info item or user to make relevant prediction

c) **Scalability:** when the volume of data is increased it may be cause the bad recommendation because computation may grow with linearly with the no of user and item.

d) **Synonymy**: It is a inclination of comparable item that have dissimilar entriesor names. Like some it is very difficult to find out the difference between closely related items like baby cloth &baby wear.

Recommender engine enable recommendations to be made to users of a system in reference to the items or elements. Currently, collaborative filtering [1] is the most commonly used and studied technology. One difficult, though common, problem for a recommender system is that most of the recommended information is concentrated in a small number of population items, which we called "Long tail phenomenon" [2][3]. Hence we proposed a detailed long tail evaluation criteria including long tail recommendation rate, long tail stability and depth which satisfies the quantitative evaluation of long tail recommendation Recommender systems enable recommendations to be made to users of a system in reference to the items or elements.

In base paper, author applied the deep learning to remove the data sparsity problem using stack auto encoder denoising technique using programmable dataset. Now, we proposed the hybrid deep learning called HDLLSTR where we combine the CNN (Convolutional neural network) and GRU (Gated recurrent units) and remove the

data sparsity problem. In our proposed the python will be a programming language and deep learning libraries like TensorFlow and Keras have used.

So getting better performance here we develop a new purposed method that is HDLLSTR which is a combination of CNN and GRU neural network

IV. Proposal Model using Hybrid Technique of Deep learning

In base paper, author applied the deep learning to remove the data sparsity problem using stack auto encoder denoising technique using programmable dataset. Now, we proposed the hybrid deep learning called HDLLSTR means Hybrid Deep learning Long Tail Recommendations system) where we combine the CNN (Convolutional neural network) and GRU (Gated recurrent units) and remove the data sparsity problem. In our proposed the python will be a programming language and deep learning libraries like TensorFlow and Keras have used.

So getting better performance here we develop a new purposed method that is HDLLSTR which is a combination of CNN and GRU neural network

V. Working of HDLLSTR :

First of all, user enter query. After getting user query, some basic preprocessing tasks has to be performed. Function relevance is applied to perform long-tail services. Pattern extraction is the next step to integrate with preference. For the extraction purpose we use novel technique named as Deep Neural Network which overcome the problem of existing techniques. Further output of this phase is passed to recommendation system and after that sorting is performed from database and create ranked list of long tail services. This completes the process of recommendation system.

a) Role of CNN In proposed Model

DNNs are hierarchical neural networks, inspired by the simple and complex cells in the human primary visual cortex. A DNN comprised of convolutional layer alternate with maxpooling layer] followed by fully connected layers and a final classification layer. DNN very definite power of learning discriminative features from raw image patches make it efficient for computer vision tasks, in comparisons to traditional handcrafting features.

CNN working is explained in following flowchart.

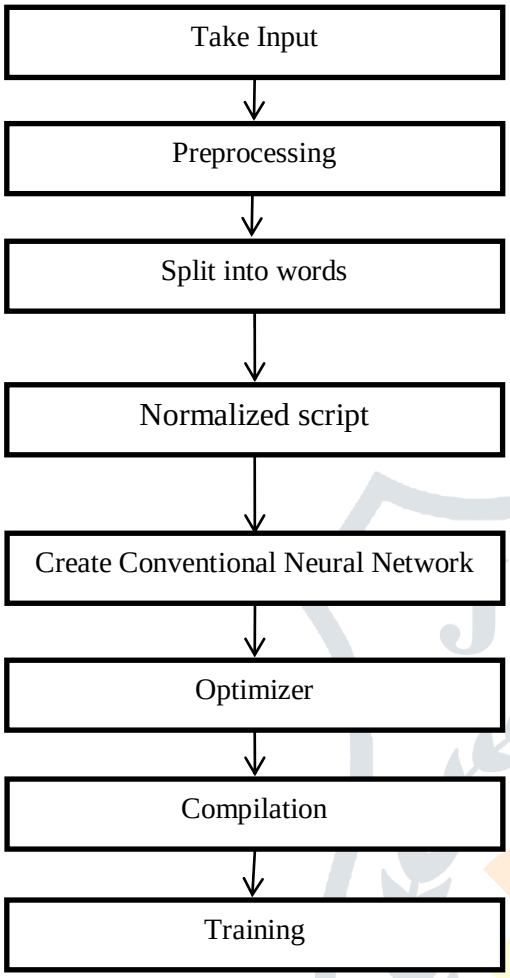


Fig.5.5 Flowchart of Neural Natworks

The network used in this work contains five layers including the classification layer; the first three are comprised of convolutional layers each followed by Maxpooling. The convolutional layers are followed by one fully connected hidden layers and the softmax classification layer is fully connected with two neurons for MA and non-MA. In this work, we have incorporated dropout [15] training algorithm for three convolutional layers and one fully connected hidden layer. And maxout activation function is used for all layers in the network except the softmax layer.CNN comes under Deep Neural Networks.

b) Role of GRU In Proposed model (Gated recurrent units**)**

GRU are a small difference on long short term memory's. They have just singleless gate and are supported somewhat in an unexpected way: rather than an info, yield and an updategate. This gate decides both how much data to keep from the last state and how much data to let in from the past layer. The reset gate worksmoresimilar to the long short term memory's forget-gate however it is positioned somewhat contrarily.

VI. Keras working for HDLLSTR by using CNN & GRU:

The main structure in Keras is the Model which defines the complete graph of a network. You can add more layers to an existing model to build a custom model that you need for your project.

Here's how to make a Sequential Model and a few commonly used layers in deep learning

1. Sequential Model

```
from keras.models import Sequential
from keras.layers import Dense, Activation,Conv2D,MaxPooling2D,Flatten,Dropout

model = Sequential()
```

2. Convolutional Layer

This is an example of convolutional layer as the input layer with the input shape of 320x320x3, with 48 filters of size 3x3 and use ReLU as an activation function.

```
input_shape=(320,320,3) #this is the input shape of an image 320x320x3
model.add(Conv2D(48, (3, 3), activation='relu', input_shape= input_shape))
```

another type is

```
model.add(Conv2D(48, (3, 3), activation='relu'))
```

3. MaxPooling Layer

To downsample the input representation, use MaxPool2d and specify the kernel size

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

4. Dense Layer

adding a Fully Connected Layer with just specifying the output Size

```
model.add(Dense(256, activation='relu'))
```

5. Dropout Layer

Adding dropout layer with 50% probability

```
model.add(Dropout(0.5))
```

Compiling, Training, and Evaluate

After we define our model, let's start to train them. It is required to compile the network first with the loss function and optimizer function. This will allow the network to change weights and minimized the loss.

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

Now to start training, use fit to fed the training and validation data to the model. This will allow you to train the network in batches and set the epochs.

```
model.fit(X_train, X_train, batch_size=32, epochs=10, validation_data=(x_val, y_val))
```

Our final step is to evaluate the model with the test data.

```
score = model.evaluate(x_test, y_test, batch_size=32)
```

Results : In this section we discuss how to built a recommendation system by using Keras tool and find out the result step by step file creation which is following .

1) Training.py: contain the main training loop.

   By using this steps in Python programming we get 97% training accuracy fig. show the result.

2) Create_model: to change the model.
3) Validation.py:it is used to check that supplies value conform to specification.
4) Accuracy .py: to check the accuracy of train and test data.
5) Classification.py: it is used to check for confusion metrix or prediction result. So by using these steps we easily evaluate our recommendation system.To check the how many true result return from total data set . We calculate precision,recall f1-score and support metrics also called (confusion matrices)by using following equation.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Support=(2*Recall*Precision)/(Recall+Presision)

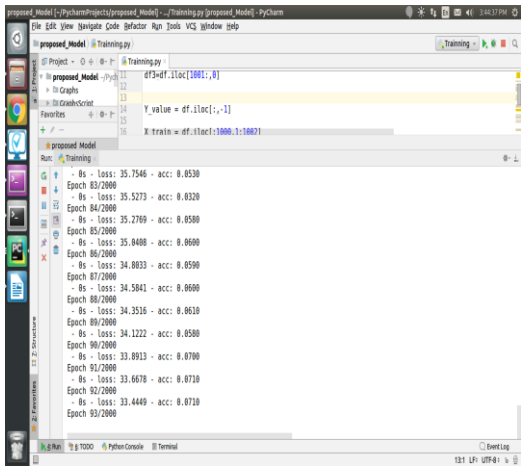


Fig .Training data

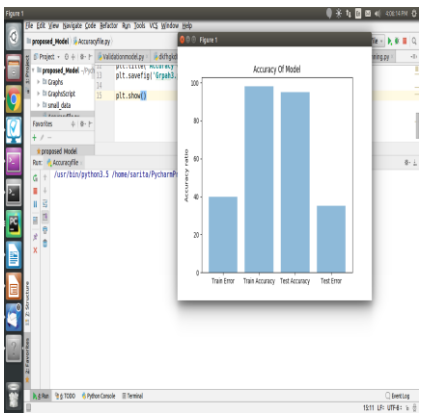


Fig .Accuracy of Train data

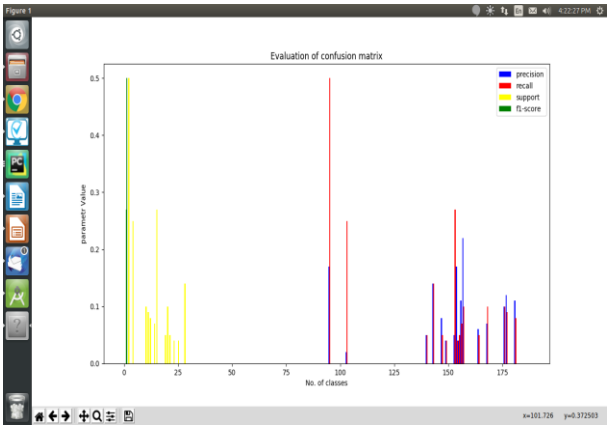


Fig. classification matrices

VII. Conclusion and future scope:

The proposed deep learning framework which is specifically designed to handle the problem of data sparsity and unmatched quality of base paper which was using by various developers. To handle the data sparsity problem, we proposed deep learning model HDLLTSR using SDAE (Stack-de-noising Auto Encoder) as simple to understand the modeling and strong effective representations. Furthermore, the information from hot services has transferred and implement as regularization of Stack-de-noising Auto Encoder output to understand the pattern. To create the usage of some long-tail historical ratings, a different technique has been designed to model the developer's preferences. The experiments defines the techniques to gain the maximum improvement composed with the state of the art baseline techniques. In the proposed method, we overcome the data sparsity with implementation of CNN and GRU and achieved the maximum accuracy of the model. . We can also think more to investigate about more Machine learning and deep learning neural networks which can be implemented like RNN, CNN with the help of some more libraries like Tensorflow, Keras, Pandas, ReLU and further to increase the performance.

VIII.REFERENCES

1. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., & Stumme, G. (2008). Tag recommendations in social bookmarking systems. *Ai Communications*, *21*(4), 231-247.
2. Oestreicher-Singer, G., & Sundararajan, A. (2012). Recommendation networks and the long tail of electronic commerce. *Mis quarterly*, 65-83.
3. Singhal, A., Sinha, P., & Pant, R. (2017). Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works. *arXiv preprint arXiv:1712.07525*.
4. Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., & Venturini, R. (2012, August). Efficient query recommendations in the long tail via center-piece subgraphs. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 345-354). ACM.

5. Celma, Ò., & Lamere, P. (2008, October). If you like the beatles you might like...: a tutorial on music recommendation. In *Proceedings of the 16th ACM international conference on Multimedia* (pp. 1157-1158). ACM.
6. Song, Y., Dixon, S., & Pearce, M. (2012, June). A survey of music recommendation systems and future perspectives. In *9th International Symposium on Computer Music Modeling and Retrieval* (Vol. 4).
7. Craw, S., Horsburgh, B., & Massie, S. (2015, September). Music recommendation: audio neighbourhoods to discover music in the long tail. In *International Conference on Case-Based Reasoning* (pp. 73-87). Springer, Cham.
8. Domingues, M. A., Gouyon, F., Jorge, A. M., Leal, J. P., Vinagre, J., Lemos, L., & Sordo, M. (2013). Combining usage and content in an online recommendation system for music in the long tail. *International Journal of Multimedia Information Retrieval*, *2*(1), 3-13.
9. Hinz, O., & Eckert, J. (2010). The impact of search and recommendation systems on sales in electronic commerce. *Business & Information Systems Engineering*, *2*(2), 67-77.
10. Hu, X., Zhang, C., Wu, M., & Zeng, Y. (2017, October). Research on Long Tail Recommendation Algorithm. In *IOP Conference Series: Materials Science and Engineering* (Vol. 261, No. 1, p. 012019). IOP Publishing.
11. Dennis, J. (2016). Search Engine Optimization and the Long Tail of Web Search.
12. Konstas, I., Stathopoulos, V., & Jose, J. M. (2009, July). On social networks and collaborative recommendation. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (pp. 195-202). ACM.
13. Lam, S. K., & Riedl, J. (2004, May). Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web* (pp. 393-402). ACM.
14. Gaffney, M., & Rafferty, P. (2009). Making the long tail visible: social networking sites and independent music discovery. *Program*, *43*(4), 375-391.
15. Oestreicher-Singer, G., & Sundararajan, A. (2012). Recommendation networks and the long tail of electronic commerce. *Mis quarterly*, 65-83.
16. Szpektor, I., Gionis, A., & Maarek, Y. (2011, March). Improving recommendation for long-tail queries via templates. In *Proceedings of the 20th international conference on World wide web* (pp. 47-56). ACM.
17. Huang, H., Zhao, Y., Yan, J., & Yang, L. (2015). Improve the 'long tail'recommendation through popularity-sensitive clustering.
18. Shi, K., & Ali, K. (2012, August). Getjar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 204-212). ACM.
19. Shi, L. (2013, October). Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach. In *Proceedings of the 7th ACM conference on Recommender systems* (pp. 57-64). ACM.

20. Huang, Z., Cautis, B., Cheng, R., Zheng, Y., Mamoulis, N., & Yan, J. (2018). Entity-Based Query Recommendation for Long-Tail Queries. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *12*(6), 64.

21. Lee, K., Yeo, W. S., & Lee, K. (2010). Music Recommendation in the Personal Long Tail: Using a Social-based Analysis of a User's Long-Tailed Listening Behavior.

22. Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the long tail recommendation. *Proceedings of the VLDB Endowment*, *5*(9), 896-907.

23. Zhou, W., Li, J., Zhang, M., Wang, Y., & Shah, F. (2018). Deep Learning Modeling for Top-N Recommendation With Interests Exploring. *IEEE Access*, *6*, 51440-51455.

24. Hui, S., Pengyu, L., & Kai, Z. (2011, August). Improving item-based collaborative filtering recommendation system with tag. In *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*(pp. 2142-2145). IEEE.