Int. J. Nonlinear Anal. Appl. 13 (2022) No. 1, 2411-2423
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.5941

Recommendation engines-neural embedding to graph-based: Techniques and evaluations

Ali Akbar[a], Parul Agarwal[a,*], Ahmed J. Obaid[b]

[a]Department of Computer science and Engineering, Jamia Hamdard, New Delhi, India
[b]Faculty of Computer Science and Mathematics, University of Kufa, Iraq

(Communicated by Madjid Eshaghi Gordji)

Abstract

The goal of any profit organization is to bolster its revenue by providing useful suggestions to its customer base. In order to achieve this, vast research is being undertaken by companies such as Netflix and Amazon on their Recommendation Systems and providing users with choices, they are most likely to click on. The purpose of this paper is to provide a holistic view of types of Recommendation Engines and how they are implemented, scaled and can provide a basis for revenue generation. The focus would be to implement a Recommendation Engine on PySpark using the ALS (Alternate Least Square) method. Besides, Neo-4j and Cypher query language for implementing recommendations on a graph database and analyzing how heterogeneous information can be levied to tackle the infamous cold start problem in recommender engines would be explored. The dataset used for analysis is the Group-lens 100K Movie-lens dataset and the algorithm is implemented to best fit the dataset. Further, an in-depth comparison of several techniques has been carried out on the basis of different metrics, hyper-parameter selection and the number of epochs used. The claims have been justified by evaluating the performance of the model depending on the different use cases, thus aiding in predictive analytics of the movie, as per the interest of the customer using visualization tools.

Keywords: Collaborative filtering, content-based filtering, popularity-based filtering, recommender system, neural networks.

*Corresponding author
Email addresses: aliakbar392@gmail.com (Ali Akbar), pagarwal@jamiahamdard.ac.in (Parul Agarwal), ahmedj.alajanaby@uokufa.edu.iq (Ahmed J. Obaid)

Received: September 2021 *Accepted:* November 2021

1. Introduction

Recommendation System (RecSys/RS) is an information filtering technique which provides users with the information he may be interested in. The primary purpose of a recommender system is to suggest items to users on the basis of his past experiences or choices. The world of RecSys (Recommendation System) has evolved exponentially since the advent of the age of Artificial Intelligence and availability of memory and computation sources. Companies that employ RecSys and leverage its benefits are YouTube, Facebook, Coursera, OTT (Over the top) streaming services such as Netflix and Prime Video and e-commerce giants Flipkart and Amazon. These businesses use RecSys in a variety of ways such as suggesting videos to watch (Netflix), products to buy (Amazon) or courses to undertake (Coursera) based on your previous actions or historical data. In this paper, we are going to analyze different types of recommendation systems, how they are created using frameworks like Pytorch and how to evaluate a successful system based on regression metrics like RMSE (Root Mean Square Error). We, will be using the famous IMDB dataset for our study and try to figure out which technique is most suitable for movie recommendations. The main contributions of this paper are:

- To understand what a recommender system is, its advantages to the business organizations, and how it works.
- To review the performance of three different techniques: PySpark, PyTorch (neural -based), and Neo-4j (graph-based) approaches for a movie-lens dataset.
- To examine these on the basis of three parameters: RMSE (Root Mean Square Error), Hyper-Parameter selection, and the number of epochs used and identify the best and average performance of these in different scenarios.

This paper is organized as follows: Section 2 throws light on the types of recommendation systems and further explores some related work. Section 3 presents an extensive review of the three techniques, namely, PySpark, Neo-4j, and graph-based systems and illustrates with results. Section 4 concludes the paper by identifying the future scope.

2. Background

Recommendation systems are categorized into four major types: Popularity-based, Content-Based, Collaborative, and Hybrid which are further explored.

2.1. Types of Recommendation Systems

2.1.1. Popularity-based RS

The assumption is to recommend the most popular item to the user. This approach is easy to implement and provides a good baseline for all other models. If there is a user on which we don't have any historical data, popularity-based approach can come up as a rational solution. It is logical for the algorithm to propose a set of items or movies which are popular and have been rated the greatest number of times. Popularity based method can sometimes be more efficient than complex Collaborative Filtering techniques. There are many baseline algorithms available for this approach which have been discussed in [6].

2.1.2. Content-based RS

The Content-based filtering (CBF) also known as cognitive filtering uses a customer's prior movie choices as information and finds items similar to what user has shown interest for in order to generate recommendations in the future. The items are grouped together based on the concept of descriptors. CBF uses a variety of models to induce similarity between items in order to generate sufficient recommendations. We can use Vector Space Model like TFIDF (Term Frequency Inverse Document Frequency) [33] and probabilistic models like Neural Networks [5] or Naive-Bayes Classifiers. It is more about the information of the products [2] such as Movie-Id, Movie Genre or the cast and directors of the movie. CBF techniques are capable of tackling the cold-start problem and if the preference of the user changes it has better scope to adjust in a less span of time. Also, it ensures privacy and security to the customer [23].

2.1.3. Collaborative Filtering based RS

The Collaborative Filtering (CF) algorithms are based on the fact that if two clients have similar content history then they will behave exactly in a similar manner in the future [3, 4, 30]. It clusters user preferences based on the similarity in their choices. Collaborative Filtering techniques work by building a database referred to as User-Item matrix or Utility Matrix for preferences for items by users. It then proceeds to calculate similarity based on the matrix obtained. This approach does not use metadata about the items or the customers. CF algorithms can be divided into 2 main categories-Memory based [15] and Model Based [11].

2.1.4. Hybrid RS

Hybrid filtering technique combines output of other techniques and stacks them with one another to provide more generalized outputs. Ensembling of different models such as CF or Content based can be viewed as a hybrid recommendation engine.

2.2. Related Work

The first paper on Recommendation System was published in the year 1992 [26] and since then it has been a source of extensive research and development. [27, 25] described recommender systems as a software tool for proposing or coming up with suggestions. Recommendations depend on user feedback which can be in the form of a review or a categorical rating from 1 to 5. Sentiment Analysis techniques can be applied to models which feature natural language and convert it to a numerical or binomial output to be fed into our machine learning architecture [12]. Weighted techniques can be used to generalize the tokenizing process [8]. The 2 most popular adaptations of Recommendation Systems are Content Based-Filtering and Collaborative Filtering. Content based filtering [10] provides a method for automatically filtering, categorizing and producing outputs in the form of recommendations to the user. This approach involves the comparison between items present in the user's basket and items which are rated in the past to provide recommendations.
In collaborative filtering [1], instead of computing user interactions with a given sample set, it is compared with the other set of users and similar user groups are identified for a generalized prediction approach are found. Researchers have proposed a hybrid recommender model to tackle the problem of cold start, which explores latent item features, learned from a deep learning neural network and extracts those engineered features to provide input to the $SVD++$ CF model [28]. Similarly, a CF algorithm using ALS as its loss and optimizing metric which deals with high dimensional sparse data by factorizing input into smaller dimensions using matrix decomposition techniques, has been proposed by [34]. An efficient way to deal with high dimensional data especially textual is to explore and come up with a technique to embed word embeddings in a 2-dimensional space and study

their interactions. This model is known as the word2vec model in machine learning literature [19]. This technique is used to supply input to the Neural Network in the form of Neural Embeddings. Machine Learning community has come up with several implementations of recommendation systems with the most recent being the use of Knowledge Graphs where the aim is to combine external information sources to provide wider extent and improve the efficiency of the collaborative filtering algorithm by extracting usable insights and make the model more adaptive and able to train on past data. Facebook developed a Deep Learning based model which was named as Deep Learning Recommendation Model (DLRM) [16], which provided a different approach of dealing with Neural Networks in its ability to handle categorical variables and open sourced its implementation on Pytorch [21, 31]. Metrics used for measuring the performance of recommendation models are divided into statistical and decision support accuracy metrics [32]. Common examples of statistical measures used are RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) and some decision support measures are computed from the confusion matrix such as Precision, F1-score and AuC-Roc curve (Area under curve Receiver operating characteristic). Thus, coverage and accuracy metrics play an important part for evaluating the performance of our system [9].
In the next section, we shall implement a Recommendation Engine on PySpark using the ALS (Alternate Least Square) method. Neo-4j and Cypher query language would be used for implementing recommendations on a graph database and it would be analyzed how heterogeneous information can be levied to tackle the infamous cold start problem in recommender engines. An analysis of the results would be conducted to understand the best technique for various parameters.

3. Methods and Results

The experimental analysis in this paper is organised as follows. We will begin by stating briefly about the technology under consideration and its basic functionalities. How it deals with big data and sparse data and why it has become a golden standard in the world of recommendation engines. The experimental approach is similar to a machine learning pipeline where we will begin by importing the data to the platform (Spark, Jupyter notebook or Neo-4j for graphs). The data will be cleaned and transformed to be suitable for our recommendation project. After the data is ready for use case experiment we will perform exploratory data analysis on it to find valuable insights and aggregate level statistics which will summarize the data we are dealing with. The data sparsity problem will be solved by doing necessary wrangling of the data. We will proceed by choosing a suitable machine learning algorithm to tackle our predictions and choose the set of hyper parameters which give us the optimum result (i.e. least error). The result will be recorded for all the 3 approaches from where we will delve into their comparisons and aim to find the most suitable approach for making recommendation engines. The Movie Lens [17] dataset would be used to benchmark performance of each evaluation method. The criterion of judging the capabilities of the methods would be RMSE. Two data frames represent the dataset we're going to analyze.
Tab. 1 depicts the rating user gave to a particular movie represented by its movieid. The mapping from movieid to movie can be achieved by taking a look at Tab. 2. Movieid 1 signifies Toy Story which was released in the year 1995. Timestamps are time-coordinates representing the time when the explicit feedback was provided by the user. It can be seen that not every user has watched all the movies so the matrix which will be highly sparse and we will need to manipulate it accordingly so that it does not yield to impractical or infeasible results. Tab. 2 describes the title and genre of the movie. Genre of a movie can be argumentative so the dataset has used Imdb (International Movie Database) genre feature and inculcated that in this relation. As we have only a few features, we will include all during our model training and validation and feature engineering is minimal. Our

Table 1: Dataframe depicting rating given by a user for a movie at a particular timestamp

Index	userId	movieid	Rating	Timestamp
0	1	1	4.0	96234235
1	1	3	4.0	961246124
2	1	6	4.0	963561245
3	1	47	5.0	962146321
4	1	50	5.0	961254643

Table 2: Dataframe mapping the movied to its title and genre

Index	movie-id	Title	Genres
0	1	Toy Story	Adventure/Animation/Children/Comedy/Fantasy
1	2	Jumanji	Adventure/Children/Fantasy
2	3	Grumpier Old Men	Comedy/Romance
3	4	Waiting to Exhale	Comedy/Drama/Romance
4	5	Father of the Bride Part 2	Comedy

aim is to segment similar users together and try to estimate the rating which a user belonging to a particular segment would give to a particular movie or a set of similar movies. If a customer has rated a movie n highly, it would naturally relate to a user being shown recommendations similar to the movie n.

Fig. 1 depicts the number of customers in the Y-axis and the rating they gave to a particular movie on the X-axis. We can see that 4 is the most common rating (Mode) given by the customers and 1 the least. By doing some basic EDA (Exploratory Data Analysis) on the given data frame we can find out that the most popular movie is Star Wars (1977). It has been rated a total of 583 times.

There is a total of 610 unique users who have rated 9724 unique movies. Basic exploratory analysis data on ratings in shown in Fig.2 depicts that the total number of items in our data is 100,000 (Ratings). Mean, Standard Deviation and Quantile values are as represented. We have used this data as the input for our model trained by Alternating Least Square method on Spark. It is important to do data preprocessing in Machine Learning as it leads to better accuracy and robust models. We will preprocess the data for supplying it as input to the Neural Network.

As neural networks require large amount of data in order to mine and find relationships between variables we select the top movies and users and generate a data frame as represented in Fig. 3.

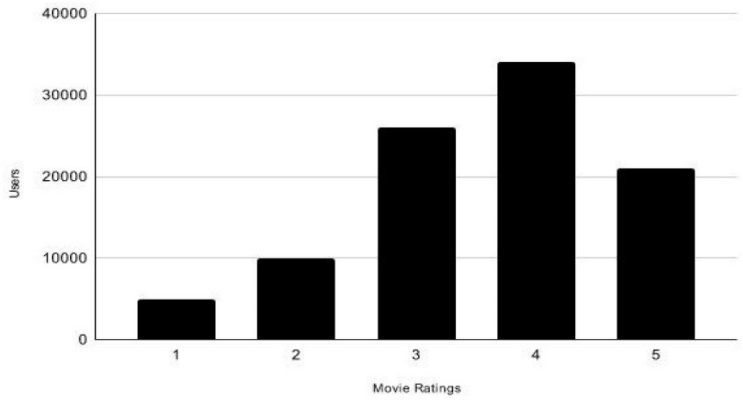

Figure 1: Depicts the number of customers and their rating for a movie.

```
count    100000.000000
mean          3.529860
std           1.125674
min           1.000000
25%           3.000000
50%           4.000000
75%           4.000000
max           5.000000
Name: rating, dtype: float64
```

Figure 2: Descriptive Statistics of the Movie Lens dataset

movieId	1	50	110	260	296	318	356	480	527	589	593	1196	2571	2858	2959
userId															
68	2.5	3.0	2.5	5.0	2.0	3.0	3.5	3.5	4.0	3.5	3.5	5.0	4.5	5.0	2.5
182	4.0	4.5	3.5	3.5	5.0	4.5	5.0	3.5	4.0	2.0	4.5	3.0	5.0	5.0	5.0
249	4.0	4.0	5.0	5.0	4.0	4.5	4.5	4.0	4.5	4.0	4.0	5.0	5.0	4.5	5.0
274	4.0	4.0	4.5	3.0	5.0	4.5	4.5	3.5	4.0	4.5	4.0	4.5	4.0	5.0	5.0
288	4.5	NaN	5.0	5.0	5.0	5.0	5.0	2.0	5.0	4.0	5.0	4.5	3.0	NaN	3.5
307	4.0	4.5	3.5	3.5	4.5	4.5	4.0	3.5	4.5	2.5	4.5	3.0	3.5	4.0	4.0
380	5.0	4.0	4.0	5.0	5.0	3.0	5.0	5.0	NaN	5.0	5.0	5.0	4.5	NaN	4.0
387	NaN	4.5	3.5	4.5	5.0	3.5	4.0	3.0	NaN	3.5	4.0	4.5	4.0	4.5	4.5
414	4.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	4.0	5.0	4.0	5.0	5.0	5.0	5.0
448	5.0	4.0	NaN	5.0	5.0	NaN	3.0	3.0	NaN	3.0	5.0	5.0	2.0	4.0	4.0

Figure 3: Dense matrix representing top movies and users

Table 3: Hyperparameters selection

Coldstart Strat	Rank	Maxiter	RegParam	Implicit Prefs	Alpha	Non-negative
Drop	10	10	.01	False	1.0	True

Table 4: Hyperparameters selection

	Training Set	CV set
Percent Split (Random)	75	25
No. of Items	75000	25000

This data frame will help us tackle the cold start, and data sparsity problems associated with Recommender Systems. The third approach involves using heterogeneous information in the form of graphs and depicting the relationships through nodes and edges. We will be using the Neo4j graph framework for generating graphs and producing recommendations. In the next three sub-sections, let us explore each of the three techniques: Apache Spark, PyTorch, and Neo-4j.

3.1. Apache Spark

Apache Spark [18] is an open sourced cluster computing framework which is general purpose and provides an interface for building and manipulating Big Data. Spark is easy to access and use, offering APIs in 4 programming languages: Python, Java, Scala, and R. Since, Sparks' machine learning library (MLLib) found widespread success with its ability to handle distributed computing and Big Data in the form of its basic abstraction RDDs (Resilient Distributed Datasets). Hence, we will implement a Recommender System on MLlib, Taking ALS (Alternate Least Squares), we fit our model by keeping one factor fixed while adjusting the other factor. This process goes on until convergence. We will implement a model-based CF technique to generate recommendations and use MLlib ALS method for training our model.
PySpark allows users to set the coldStart Strategy parameter to "drop" which drops rows in the DataFrame of predictions that contain Null or NaN values. The evaluation metric will then be computed over the non-NaN normalized data and will be valid for inferences and hypothesis testing. We will train the model on the hyper parameters stated in Tab. 3. The rank parameter is representative of the number of latent factors in the model. Rank will represent the number of categories in the data. The default value is 10 which we will continue to use for our model. As we are dealing with big data, it is imperative to define a MaxIter parameter which makes sure we don't run out of memory and its default value is 10. It means that we will run a maximum of 10 iterations on the data sample. RegParam is the regularization parameter. $implicitPrefs$ is set to False which specifies that we are using explicit data for our predictions. Parameter alpha is responsible for generating a baseline model on which our predictions can be judged. Non-negative is set to True to signify we are using nonnegative values for generating least squares.

The training-validation split used for our model is 75-25. As can be seen in Tab. 4, 75,000 rows are used for training the model and learn the patterns in data whereas we will be using our cross-validation data which is unseen to our model for evaluation and error calculation.

Fig. 4 depicts the predictions by our model for individual items. RMSE of 0.62342 is observed for the model.

3.2. Neural Embeddings and Pytorch

Pytorch is a framework that allows us to build various computational graphs and run them on GPU (Graphical Processing Unit) for greater performance and less execution time. We will

```
root
 |-- userId: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- rating: integer (nullable = true)
 |-- title_new: double (nullable = false)
 |-- prediction: float (nullable = false)

+------+--------------------+------+---------+----------+
|userId|               title|rating|title_new|prediction|
+------+--------------------+------+---------+----------+
|   456|       Loaded (1994)|     4|   1293.0| 2.5004873|
|   561|Victor/Victoria (...|     3|    430.0| 2.5559244|
|   640|    Crow, The (1994)|     4|    233.0|  4.162498|
|   234|  Primal Fear (1996)|     3|    145.0| 3.0594409|
|   505|        Alien (1979)|     3|     44.0| 2.8870814|
|    22|Three Musketeers,...|     4|    378.0| 3.0135791|
|   102|Big Blue, The (Gr...|     3|    981.0| 2.7114658|
|   428|Excess Baggage (1...|     5|    591.0| 3.4902601|
|   315|   Sting, The (1973)|     4|     75.0|  4.010038|
|   387| Pulp Fiction (1994)|     5|     12.0| 4.7546453|
+------+--------------------+------+---------+----------+
only showing top 10 rows
```

Figure 4: Prediction using ALS

```
tensor([[ 224,  472],
        [ 532, 3638],
        [ 124, 4813],
        [ 262,  348]])
tensor([[3.],
        [5.],
        [2.],
        [5.]])
```

Figure 5: X-batch and Y-batch for batch size=4

cluster similar users together using appropriate clustering algorithms and try to provide aggregated recommendations. Embeddings can also be used as intermediates in ensemble algorithms which will make it easier for our model to find correlated features. We will implement our model on Pytorch and take advantage of its GPU-oriented techniques. We will convert our arrays to high dimensional tensors for efficient utilization of GPU memory and faster computations. Review Iterator with a batch size of 32 is taken for training the model. We will not use larger batch size due to computation limits. A sample tensor is provided in Fig. 5.

Some important hyper-parameters which we should tune before acquiring results are batch size, regularization technique (Dropout would be the most suitable), no. of layers in our deep neural network, optimizers such as SGD (Stochastic Gradient Descent) plus momentum or Adam (Adaptive Moments), no. of layers and neurons per layer and determining how deep our model will be i.e. no. of hidden layers. The hyper parameters selected for our neural network are provided in Tab.5. Learning rate is selected as 10^{-3} in order to make sure our Adam [14, 7] optimizer is not stuck at local minima. Batch size of 200 is selected and patience parameter is used in case of early stopping (finding optimum value in the middle of epochs). Tab. 6 depicts the training and validation losses over 5 epochs. RMSE for this model is 0.62342.

Table 5: Hyper-parameters Selection for neural net

Learning Rate	Epoch	Loss	Optimization	Batch Size	Patience	Best loss	Best weights
10^{-3}	5	MSE loss	Adam	200	10	npi.inf	None

Table 6: Training and validation Losses over 5 epochs

epoch	train_loss	valid_loss
1	0.923901	0.946068
2	0.865458	0.890646
3	0.787896	0.836753
4	0.638374	0.815428
5	0.561979	0.814652

3.3. Graph Based Recommendations using Neo-4j and Cypher Query Language

A graph database, models and stores data as nodes and edges of a graph structure [22]. Graph databases allow methodical and rapid information retrieval of highly connected and complex hierarchical structures that would be extremely inefficient to model using traditional relational systems. [29] put forwards a recommendation system based on similarity graphs. All the traditional algorithms of collaborative filtering, popularity approach or content- based filtering can be levied upon graph-based databases. Neo4j [24, 20] is a popular graph- based database offering multiple functionalities for generating real time graphical models. It is equipped with handling basic ACID transactions and basic data storage, organization and pre-processing. The language that is used to query graph databases in Neo-4j, is termed as cypher query language. Neo-4j utilizes property graph model where we can assign weights and labels to nodes and their relationship. Fig. 6 describes a typical graph database with nodes acting as data points and edges depicting the relationship between connected nodes. Labels are used for identifying nodes and we can have multiple labelled nodes in our database specifying its type. Two nodes are connected by their relationship. Mapping of key to value pairs can be preserved on nodes and edges.

Tab. 7 shows the predictions made by our graph database for the specific user (Movies similar to "Inception"). The genres of Inception are (Crime, Drama, Mystery, Sci-Fi, Thriller, IMAX, Action). Whereas, the attribute s1 denotes the genres of movies found similar to Inception based on Jaccard Similarity.

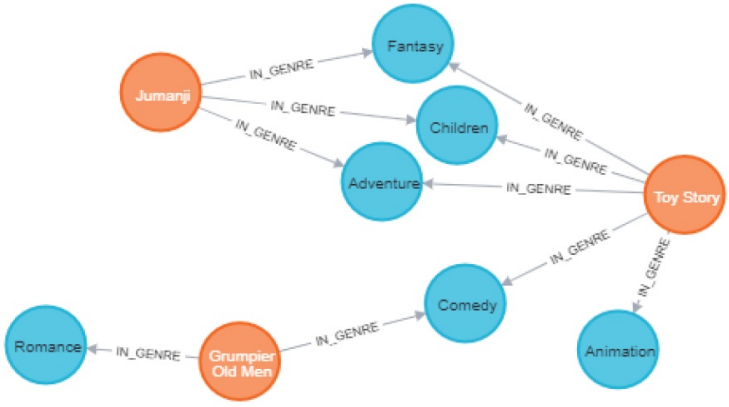

Figure 6: Movie & Genre relationship

Table 7: Recommended Movies (Similar to Inception)

Similar Movies	s2	Jaccard
Strange Days	[Crime, Action, Thriller, Sci-Fi, Mystery, Drama]	0.857143
Watchmen	[Drama, Action, Sci-Fi, Mystery, IMAX, Thriller]	0.857143
Sherlock: The Abominable Bride	[Thriller, Crime, Action, Mystery, Drama]	0.714286
Minority Report	[Crime, Mystery, Action, Sci-Fi, Thriller]	0.714286
Source Code	[Thriller, Sci-Fi, Mystery, Drama, Action]	0.714286
RoboCop	[Thriller, Sci-Fi, Drama, Crime, Action]	0.714286
RoboCop 3	[Thriller, Crime, Action, Sci-Fi, Drama]	0.714286
X-Files: Fight the Future	[Action, Crime, Mystery, Sci-Fi, Thriller]	0.714286
Man on Fire	[Thriller, Drama, Mystery, Crime, Action]	0.714286

Table 8: Comparison of PySpark, Neural embeddings and Graph-based recommendation techniques

Parameters/ Approach	PySpark	Neural Embeddings	Graph-Based
RMSE	0.62342	0.515234	0.551298(For userId=4)
Hyper-paramater Selection	Manual	Manual and resource intensive	Node and relationship to be specified
Number of epochs used	10 (MaxIter paramater)	5	None

Thus, Tab. 7 clearly shows that graph-based approach produces effectiveness in building a recommendation engine.

3.4. Results

In the end, all the 3 approaches: PySpark, Neo-4j, and Neural embedding, give unique intuitions about data. Tab. 8 illustrates the same using three parameters, RMSE, selection of Hyperparameters, and number of epochs used on a single dataset and with same amount of computing resources available for execution of tasks. In terms of accuracy achieved, Neural Networks based approaches gave the most desirable results. Contrastingly, the graph based approach is highly interpretable and can be used for analytical tasks and optimize business operations. Since, Graph based recommendations require the node and relationship representation, so are interactive and provides a clarity in identifying relationships between unrelated items. They prove to be an effective means of tracing the pattern and behavior of individuals. Pyspark can help us work in the Big Data environment by horizontally scaling our computing clusters, it provides us with a framework to do multiple tasks such as data engineering and model deployment thereby reducing our dependency and workload.

Fig. 7 is illustrative of the performance of all the 3 approaches on the basis of RMSE. Neural embedding approach gave us the RMSE of 0.515234, the least if compared to the other two. Its performance is boosted by fine-tuning model parameters and minimizing our target metric. We can achieve even better results by having better resources and compute instances. The RMSE from the PySpark approach came out to be 0.62341, after running the algorithm through 10 epochs. We can improve the results by running more iterations and selecting hyperparameters manually instead of using out of the box parameters.

The RMSE [13] for Graph-Based approach was 0.551298. Thus, we can say that the usage of a particular algorithm will depend highly on our use case. Usually if we want to make our findings more interpretable we can use Neo-4j and its graph capabilities whereas if we want to focus on building

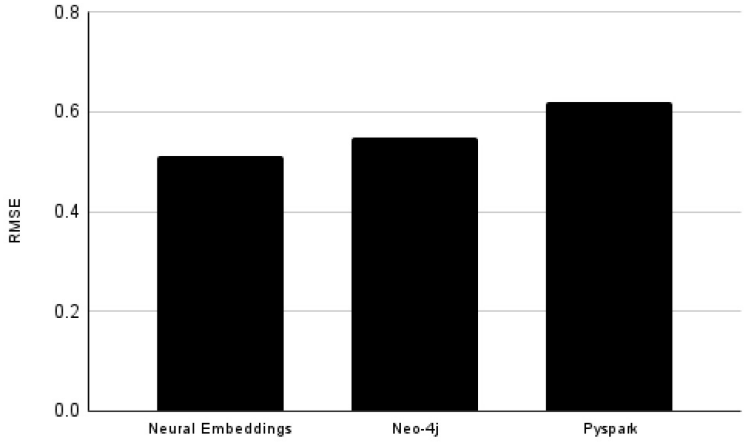

Figure 7: Comparison of PySpark, Neo-4j, and Neural-embeddings on their RMSE scores

a whole pipeline we can use Pyspark as it provides us with end to end machine learning solutions. Neural Embedding approach is still in its nascent state but the possibilities are limitless given the speed of innovation in deep learning and artificial intelligence algorithms.

4. Conclusion

Recommendation Engines have been an area of research since the early 2000s and so far, researchers have come up with several successful iterations of them. Big data technologies like Spark & Hadoop have bolstered the capabilities of these systems by providing a distributed way to handle large amounts of zeros (highly sparse) that are typically present in a utility matrix or dataset which is to be used for recommendations. Originally, it was objected that neural network would be inefficient for recommendations, but over time we found a way to input features in the form of embeddings to a neural network and generate recommendations. Facebook open sourced its neural network- based recommendation engine DLRM and is constantly working to improve its accuracy and provide state of the art results. As, we have already discussed that recommendation engines have a massive business impact and can drive the success of a product, it is mandated for a company to invest in its recommendation system and provide best results on user queries. Using graph-based recommendations have produced great results on live streaming queries by utilizing information not evident from the database, inculcating heterogeneous information about the product have led to even better results. Overall, with techniques mentioned in this paper, we will be able to develop a scalable recommendation engine which can be used for increasing revenues of a business and providing accurate predictions for user queries. However, we are very far from an ideal system because predicting a user's personality based on what he clicks or rates is a very complex task. Humans tend to be unpredictable at moments and rarely operate with fixed patterns. Choices are based on many factors outside of what a computer can predict such as historical, psychological or environmental. Creating a cognitive system is what AI industry is currently working on and if we are able to completely emulate the human mind, would result in unprecedented developments in the fields of deep learning and counterfactual machine learning. With improved quality and quantity of data and ability to use ensemble methods (3-4 models stacked together) to train our neural network so as to generalize the human behavior is the ultimate task of a recommender system.

Funding Statement

"The author(s) received no specific funding for this study."

Conflicts of Interest

"The authors declare that they have no conflicts of interest to report regarding the present study."

References

[1] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender system: A survey of the state-of-the-art and possible extensions*, IEEE Trans. Knowledge and Data Engin. 17(6) (2005) 734–749.
[2] C.C. Aggarwal, *Recommender Systems: The Textbook*, Cham. Springer Publishing Company, 1, 2016.
[3] W. Ali, S.U. Din, A.A. Khan, S. Tumrani, X. Wang and J. Shao, *Context-aware collaborative filtering framework for rating prediction based on novel similarity estimation*, Comput. Mater. Continua 63(2) (2020) 1065–1078.
[4] S. Bhatia, R. Madan, S.L. Yadav and K.K. Bhatia, *An algorithmic approach based on principal component analysis for aspect-based opinion summarization*, Int. Conf. Comput. Sustainable Global Develop. (2019) 874–879.
[5] C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 1st (reprint) ed., New York, 2006.
[6] M. Bressan, S. Leucci, A. Panconesi, P. Raghavan and E. Terolli, *The limits of popularity-based recommendations, and the role of social ties*, Int. Conf. Knowledge Discovery and Data Mining (2016) 745–754.
[7] M. Ebady Manaa, A.J. Obaid and M.H. Dosh, *Unsupervised approach for email spam filtering using data mining*, EAI Endorsed Trans. Energy Web 8(36) (2021).
[8] Z. Gulzar, A.A. Leema and G. Deepak, *PCRS: personalized course recommender systems based on hybrid approach*, Proceedia Computer Sci. 125 (2018) 518–524.
[9] A. Gunawardana and G. Shani, *Evaluating Recommender Systems*, Recommender Systems Handbook, Boston, MA, Springer, 2015.
[10] I. Hariyale and M.M. Raghuwanshi, *Design of recommender system using content based filtering and collaborative filtering technique: a comparative study*, Int. J. Adv. Sci. Tech. 29(05) (2020) 4852–4865.
[11] F.O. Isinkaye, Y.O. Folajimi and B.A. Ojokoh, *Recommendation systems: principles, methods and evaluation*, Egyptian Inf.J. 16(3) (2015) 261–273.
[12] A. Kennedy and D. Inkpen, *Sentiment classification of movie and product reviews using contextual valence shifters*, Comput. Intell. 22(2) (2006) 110–125.
[13] S.S. Khatri, D. Singh, B. Narain, S. Bhatia, M.T. Quasim and G.R. Sinha, *An empirical analysis of machine learning algorithms for crime prediction using stacked generalization: an ensemble approach*, IEEE Access 9 (2021) 67488–67500.
[14] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, The 3rd Int. Conf. Learning Represent. ICLR 2015, (2015).
[15] X. Liang, X. Zhonghang, P. Liping, L. Zhang, H. Zhang, *Measure prediction capability of data for collaborative filtering*, Knowledge and Inf. Syst. 49(3) (2016) 975–1004.
[16] N. Maxim, D. Mudigere, H.J.M. Shi, J. Huang and N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A.G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong and M. Smelyanskiy, *Deep learning recommendation model for personalization and recommendation systems*, Conf. Workshop on Neural Inf. Proc. Syst. (2019) 1–10.
[17] F.H. Maxwell and J.A. Konstan, *The movielens datasets: history and context*, ACM Trans.Interact.Intell. Syst. 5(4) (2016) 1–19.
[18] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D.B. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M.J. Franklin, R. Zadeh, M. Zaharia and A. Talwalkar, *Mllib: Machine learning in apache spark*, J. Machine Learn. Res. 17(1) (2016) 1235–1241.
[19] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, *Distributed representations of words and phrases and their compositionality*, Adv. Neural Inf. Proc. Syst. (2013) 3111–3119.
[20] A.J. Obaid, K.A. Alghurabi, S.A.K. Albermany and S. Sharma, *Improving Extreme Learning Machine Accuracy Utilizing Genetic Algorithm for Intrusion Detection Purposes*, In: R. Kumar, N.H. Quang, V.K. Solanki, M. Cardona and P.K. Pattnaik (eds), Research in Intelligent and Computing in Engineering, Adv. Intell. Syst. Comput. 1254 (2021).
[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVit, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, *Automatic differentiation in pytorch*, Proc. Neural Inf. Proces. Syst. (2017) 1–4.

[22] N.S. Patil, P. Kiran, N.P. Kavya and K.M. Patel, *A survey on graph database management techniques for huge unstructured data*, Int. J. Elect. Comput. Engin. 8(2) (2018) 1140–1149.
[23] S. Puglisi, J.P. Arnau, J. Forné and D. R. Monedero, *On content-based recommendation and user privacy in social-tagging systems*, Computer Standards & Interfaces, 41 (2015) 17–27.
[24] S. Sen, A. Mehta, R. Ganguli and S. Sen, *Recommendation of influenced products using association rule mining: neo4j as a case study*, SN Compu. Sci., 2(2) (2021) 1–17.
[25] A. Sharaff and M. Choudhary, *Comparative analysis of various stock prediction techniques*, Int. Conf. Trends Elect. Inf. (2018) 735–738.
[26] A. Sharaff and U. Srinivasarao, *Towards classification of email through selection of informative features*, Int. Conf. Power, Control Comput.Technol. (2020) 316–320.
[27] P.K. Singh, P.K.D. Pramanik, A.K. Dey and P. Choudhury, *Recommender systems: an overview, research trends, and future directions*, Int. J.Business Syst.Res. 15(1) (2021) 14–52.
[28] J. Wei, J. He, K. Chen, Y. Zhou and Z. Tang, *Collaborative filtering and deep learning based recommendation system for cold start items*, Expert Syst. Appl. 69 (2017) 29–39.
[29] L. Wu, Q. Liu, E. Chen, N.J. Yuan, G. Guo and X. Xie, *Relevance meets coverage: a unified framework to generate diversified recommendations*, ACM Trans. Intell. Syst.Technol. 7(3) (2016) 1–30.
[30] G. Xu, T. Zhijing, M. Chuang, L. Yanbing, D. Mahmoud, *A collaborative filtering recommendation algorithm based on user confidence and time context*, J. Elect.Comput. Engin. 2019 (2019) 1–12.
[31] J. Yangqing, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, *Caffe: convolutional architecture for fast feature embedding*, Int. Conf. Multimedia (2014) 675–678.
[32] D. Yashar, A. Bellogin and T.D. Noia, *Explaining recommender systems fairness and accuracy through the lens of data characteristics*, Inf. Proces. Manag. 58(5) (2021) 102662–102686.
[33] R.B. Yates and B.R. Neto, *Modern Information Retrieval*, ACM press, New York, 1999.
[34] Z. Yunhong, D. Wilkinson, R. Schreiber and R. Pan, *Large-scale parallel collaborative filtering for the netflix prize*, Proc. Algorithmic Appl. Manag.(2008) 337–348.