Greeshma Seetharaman Menon

Fashion AI Recommenders Using Deep Learning

ABSTRACT

With the rapid expansion of ecommerce platforms, fashion websites and social networks, clothing recommendation has gathered more and more interest from organizations, researchers and users alike. With increasing choice, having the right recommendation of products is very important to allow for customers to make a purchase within the platform. In this paper, we will look at implicit and explicit collaborative filtering approaches and identify whether adding features related to the products actually improves the performance of the recommender. Additionally, state of the art deep learning models in the space of recommender engines will also be evaluated. We comprehensively evaluate the performance of the models across implicit and explicit models to answer three questions – (1) Do implicit recommender engines in the space of fashion AI outperform explicit recommender engines? (2) Do hybrid recommendation engines outperform deep learning or vanilla matrix factorization methods? and (3) Do deep learning based matrix factorization approaches provide significant improvement over state of the art vanilla matrix factorization approaches? From our experiments, we have seen that pure collaborative filtering outperforms deep learning model architectures and hybrid models when the session lengths are small and there are no historical features that can be leveraged for session recommendations, thereby relying majorly on in-session recommendation to make a prediction. We also pave the way forward towards more research in the space of deep learning using alternate techniques and architectures to solve the problem.

1. INTRODUCTION

E-commerce is growing at an unprecedented rate and the fashion industry has recently witnessed a noticeable shift in customers' order behavior towards stronger online shopping. The retail fashion industry is already a big economy driver in the global landscape with ecommerce platforms such as Zalando, Walmart, Amazon etc. available for customers to purchase fashion apparels. The growth is partially fueled by the increase in the number of sellers who are able to remotely sell on the platforms, resulting in a sheer increase in variety and choice in products that can be sold to customers. Another reason for growth is the cost-effective nature for ecommerce platforms to store and deliver a huge inventory of products based on diverse customer choices that was not possible in the erstwhile brick and mortar retail stores. In spite of the positives mentioned above and the clear shift in customer preferences to purchase fashion online, there are inherent challenges with this approach, the key one being that there are too many choices available that may not be suitable to the customer who is viewing the portal. Hence, one of the key challenges for fashion platforms is to match customer preferences with the available products based on their preferences.

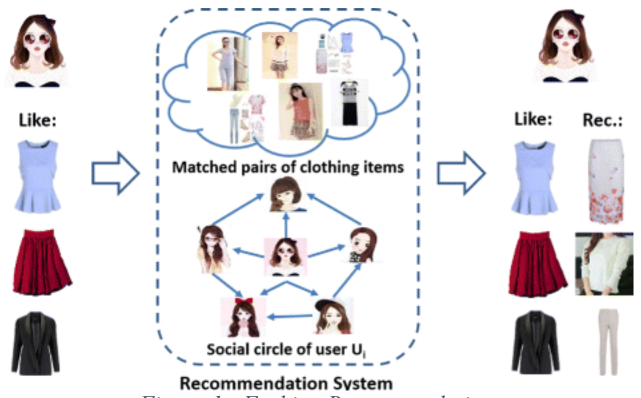

Figure 1 : Fashion Recommendation

[1]Dressipi Platform - https://dressipi.com/

This is important as ecommerce platforms do not want to inundate the customers with irrelevant products as customers have a limited attention span so that they can make purchases within their platform. Therefore, making recommendations that can convert the customer from browsing the platform to making a purchase is an important problem to solve for the ecommerce platforms.*[14]*

In this paper, we will be building a recommendation engine that can provide fashion recommendations for customers of the Dressipi[1] platform. Dressipi is a Fashion AI platform that helps fashion retailers to provide relevant products and inspiration to visitors of the retailers. The platform aims to transform how retailers engage with their customers through deeper, entirely personalized experiences through the use of data to advance the buying and merchandising processes. The data for the paper will be sourced via Recsys challenge 2022[2], the 2022 edition of ACM conference on recommendation engines organized by Dressipi, Bruce Ferwerda (Jönköping University, Sweden), Saikishore Kalloori (ETH Zürich, Switzerland), and Abhishek Srivastava (IIM Jammu, India) with the aim to ensure customer preferences are baked into building recommendations of products in a market where there is ample customization, choice and preferences.

The research hypothesis is to be able to generalize the behavior and model the customer preferences through interactions with items and provide recommendations to the current session on what products are likely to be purchased. Evaluation will be done by minimizing the loss functions between the predicted and actual scores for each interaction and identifying mean reciprocal rank (MRR) of successful recommendations to understand effectiveness of session recommendations.

The contributions done by the project will be Applied in nature. We will be leveraging existing frameworks in the Deep Learning domain such as NeuCF and comparing the performance against more traditional methodologies such as Alternating Least Squares (ALS) or Matrix Factorization approach to understand the performance of deep learning architectures on sparse datasets such as this one. The goal will be to identify areas of improvement that will be taken up as opportunities for further research to further the space of recommendation engines using Neural Network based architectures.

2. PROBLEM FORMULATION

In order to provide a recommendation of the product to show to the customer to ordain a purchase, the problem statement needs to be rephrased as a collaborative filtering based recommendation problem. In order to formulate the problem statement as the above, we will attribute a rating based on the affinity of purchase. We will be considering both explicit and implicit collaborative filtering models and create the rating of a product within a session on the basis of the following - – (1) if the product is not viewed, it gets zero rating; (2) the product with session views gets rating equal to the number of times it has been viewed in that session(example : 1,2,3 etc.) ; and (3) the product that has been purchased in the session will get a high rating associated with it (example: 10 or 100). Thus, with this transformation, we convert the problem as a matrix factorization problem that uses the session-product rating matrix to understand the session preferences and item preferences on latent factors for each session.

Considering there are m sessions, we have the vector $s = [s^0, s^1, s^2 \dots s^m]$ represent the total sessions that are done in the time period t under consideration. During the same period t, we have n items within our application and are represented by the vector $i = [i^0, i^1, i^2 \dots i^n]$.

When the system is modeled as an explicit recommendation system, products not purchased get zero ratings and products purchased are given a score of 10 or 100 based on the experiment. We attempt to provide different values for scores when type = purchase to see if this affects the decision boundary and hence, better recommendations for the session. Mathematically,

$$\forall s, i \quad Score_{s,i} = \begin{cases} 10 \ or \ 100 & when \ type_{s,i} = purchase \\ 0 & when \ type_{s,i} \neq purchase \end{cases}$$

When the system is modeled as an implicit recommendation engine, products not purchased get scores according to the number of views they have received. If an item did not get a view, it will have score = 0. When purchase has happened, the score of 10 or 100 is attributed to the score. Similar to explicit models, we attempt to provide different values for scores when type=purchase to see the effect on the decision boundary and understand if it affects recommendations for the sessions. Mathematically,

$$\forall s,i \ \ Score_{s,i} = \begin{cases} 10 \ or \ 100 & when \ type_{s,i} = purchase \\ Count(Views) & when \ type_{s,i} \neq purchase \end{cases}$$

To solve the problem of matrix factorization, we will convert the above sessions and items into a session-item interaction matrix $\boldsymbol{R}$. In order to predict the missing values of a session within a high-dimensional matrix $\boldsymbol{R}$ of size $m \times n$, we will decompose the matrix into low dimensional sessions and item matrices of rank d. This can be represented as follows

$$\boldsymbol{R} \approx \boldsymbol{S^T I}$$

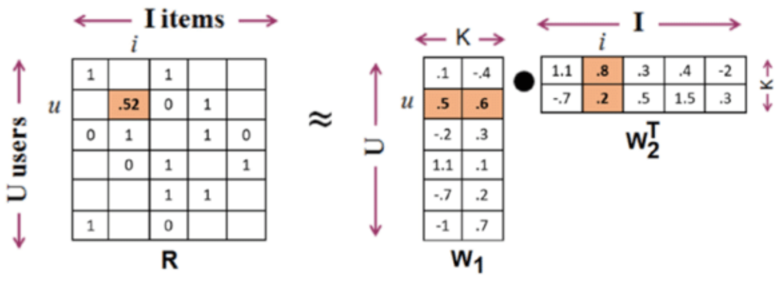

Figure 2 : Matrix Factorization Illustration

Where $\boldsymbol{S}$ *is* of size $m \times d$ and $\boldsymbol{I}$ is of size $d \times n$ and $d < \min(m, n)$. As part of our problem, we will aim to extract the values of S and I matrices that minimize the regularized mean squared error loss defined below.

$$\min_{S,I} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(R_{i,j} - S_i^T I_j\right)^2 + \frac{\alpha}{2} ||S||^2 + \frac{\beta}{2} ||I||^2$$

In our paper, we will evaluate state of the art matrix factorization approaches, hybrid approaches and neural network architectures to understand the efficacy of deep learning based architectures in modeling complexity of the matrix factorization problem for this problem. We will solve for the cold start problem [3] for estimating recommendations for new sessions by using nearest neighbor based approaches (K-NN) using cosine and Euclidean distances and understanding the recommendations based on weighted distance from the query session.

3. RELATED WORK

The need for recommendation engines within fashion industry is well documented in the literature *[3,11,12]*. Koren et al., in their paper, showcase the effectiveness of matrix factorization methods when it comes to collaborative filtering based recommendation systems. Similarly, there are several papers *[1,8]* that have explored the use of implicit vs explicit feedback within the collaborative filtering context. In the paper, the authors propose changes to the traditional ways of computing Singular Value Decomposition(SVD) and applying it to the user-item rating matrix. One modification over the existing SVD based factorization is to reduce the regularized squared error(L2 norm) to avoid overfitting rather than perform imputation or complex missing value treatment to create representative dense vectors. The limitation of this paper is the scalability of the approach as it was not parallelizable and doesn't allow

[2] https://www.recsyschallenge.com/2022/

the capability to introduce other features to finetune performance. With large datasets that are typically produced in ecommerce platforms by users, this approach can face challenges in scaling the matrix factorization problem. In order to overcome the limitation with parallelizability and ability to scale to large datasets, Y.Zhou et al. proposed the alternating least squares algorithm that can perform matrix factorization in parallel. This approach enables the dataset to be scaled to larger interaction matrices. However, the key limitation of this approach is the inability to bring additional features that can be used to finetune performance. However, when we have a cold start problem (the condition where there are high number of new sessions/users and items), M.Kula proposes a hybrid approach that leverages content based features within the matrix factorization framework where the session features are a linear combination of the features' latent representation. This approach, termed as LightFM, is an important work that is used across fashion organizations such as Lyst to solve a similar fashion recommendation problem. Latest research in the space of matrix factorization is by leveraging neural network architectures to learn complex non-linear relationships between the sessions and items to minimize the L2 norm of the regularized mean squared error. Neural Collaborative Filtering, proposed by X.He et al. is a promising aspect where session and item features can be mapped to an embedding space that can be trained by Multi-Layer Perceptron (MLP) and Generalized Matrix Factorization Layers (GMF).

In the problem formulation section, we will be formulating the problem as a matrix factorization problem that uses either explicit or implicit feedback based on purchase and view patterns. The contributions done by the paper will be to answer three questions that will be discussed in detail in the metrics and methods section – (1) Does implicit recommender engines in the space of fashion AI outperform explicit recommender engines? ; (2) Does hybrid recommendation engines such as LightFM outperform deep learning or vanilla matrix factorization methods? and ; (3) Does deep learning based matrix factorization approaches provide significant improvement over state of the art vanilla matrix factorization approaches? . Through the three questions discussed, we will be able to ascertain next steps in designing Fashion AI Recommendation Engines.

4. DATASET

The datasets are part of the Recsys 2022 challenge[2] . The datasets consist of 1.1 million retail transactions that resulted in a purchase within the Dressipi application. On average, 51% of the total visitors to the Dressipi application are new and hence no historical data is relevant to the current session. For existing users, the preferences change rapidly and hence, the whole problem is formulated in the above section as session-item interactions as recommendations need to be made based only on the current session activity. The interaction matrices are expected to be sparse in nature as the sessions themselves are short in length (session length < 3 for 50% of the transactions)

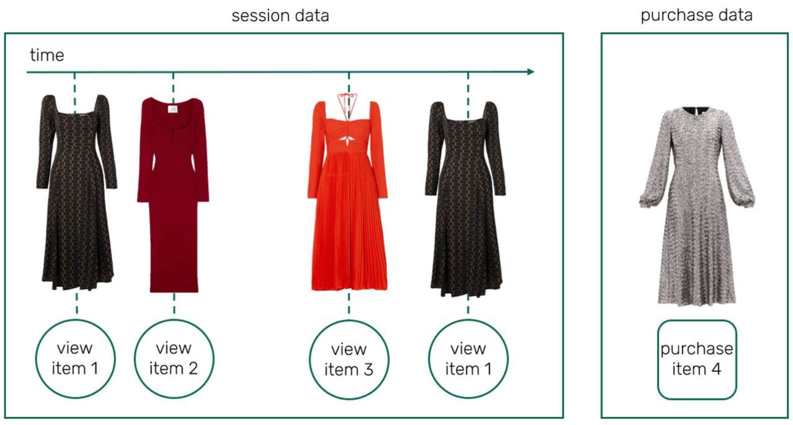

Figure 3: Sessions and Purchases Dataset Explanation

In addition to the transactions, all items in the dataset are provided with item content data, where feature related information is extracted from the products that are present across the products. The content data consists of descriptive elements of items (apparel) such as color, length, neckline, sleeve type etc. that have been manually annotated using fashion expert labeling (human in the loop reviews). However, the entire features are anonymized and cannot be made sense by looking at it visually nor additional information provided on the nature of the features.

The datasets consist of the following:

- **Sessions**: The items that were viewed in a session. In this dataset a session is equal to a day, so a session is one user's activity on one day.
- **Purchases**: The purchase that happened at the end of the session. One purchased item per session.
- **Item features**: The label data of items. Things like "color: green," "neckline: V-neck," etc. are represented within the dataset but are anonymized. For example, color: green could have been anonymized as feature category = 583 and feature value = 100 (e.g. : to represent feature category 583 has value 100 or color has value green). However, the definitions are not provided as part of the dataset and hence cannot be understood. There are totally 500 unique features and values within the dataset.

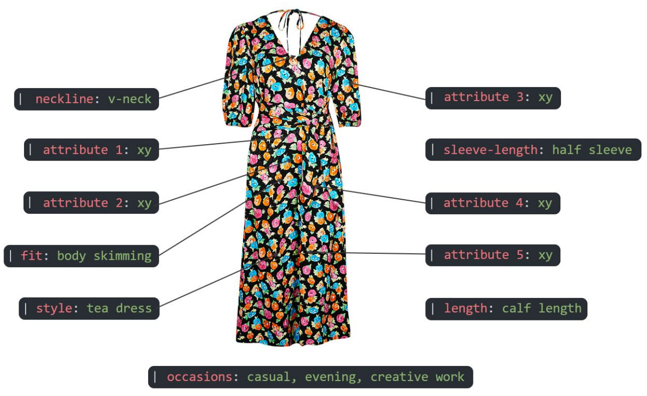

Figure 4 : Item Features dataset explanation

Additional information on the dataset can be found below :

- There are no image data available in the dataset or additional information about the item or session.
- Content data (garment labels data) is supplied for all items in the dataset. Some candidate items might not have any data in the training sessions or purchases but they will have content data.
- Content data is representative of a "category: value" taxonomy e.g., "color: blue" or "neckline: V-neck". Most feature categories will only have one value for a garment, however, there are some that have multiple values. For example, an item might have both "secondary_color: black" and "secondary_color: white". In these cases, an item will have two or more entries (rows) with the same category id and different value ids.
- Some items may not share any feature_category_ids if they are different types of items, for example trousers will share very few category ids with shirts and will share even less category ids with sandals.
- Items will have a different number of label assignments depending on how complicated they are. A basic black t-shirt will have less feature category ids, and thus less rows in the content data, compared to an evening dress with intricate details.
- Sessions are anonymous, there is a session_id but no user_id, so you won't know if two sessions are by the same person.
- The dataset only contains one purchased item per order (chosen at random). This means a session might have resulted in the purchase of a shirt and a trouser but in the dataset you can

[2] https://www.recsyschallenge.com/2022/

only see the shirt purchase. This is a limitation but the size of the dataset should be sufficiently large to compensate for it.

5. EXPERIMENT METHODS

We will be considering 6 techniques across the space of matrix factorization, deep learning and hybrid collaborative filtering in order to answer the following three questions –

1. Do implicit recommender engines in the space of fashion AI outperform explicit recommender engines?

2. Do hybrid recommendation engines such as LightFM outperform deep learning or vanilla matrix factorization methods?

3. Do deep learning based matrix factorization approaches provide significant improvement over state of the art vanilla matrix factorization approaches?

In the following section, I will be providing details on the experimentation itself and some key parameters that are used to finetune the model, thereby allowing reproducibility in the whole setup. Recommendations are provided by averaging scores of the 3 nearest neighbors to a session. Nearest neighbor calculated using K-NN algorithm with k = 3 across the session latent space.

5.1. Bayesian Personalized Ranking (Matrix Factorization) Algorithm

The Bayesian Personalized Ranking (BPR) algorithm is a state of the art matrix factorization algorithm proposed by Rendle et al., (2009). It is widely used across recommendation engines. In this particular loss function, the problem is treated as a Bayesian problem and performs pairwise training consisting of both positive and negative pairs (missing values), where the aim of the recommender engine is to maximize the posterior probability of the distribution to match the positive and negative pairs observed[15]. We leveraged the implementation by the LightFM library for this method to train the model. We considered different number of latent factors with the optimal number of latent factors converging at n = 50. Additional parameters such as epochs etc. were kept at 3000 and learning rates were automatically determined through the adagrad learning rate schedule (as it showed superior performed over adadelta).

5.2.Weighted Approximate-Rank Pairwise (Matrix Factorization) Algorithm

The Weighted Approximate Rank algorithm is another state of the art matrix factorization algorithm proposed by Weston et al. that is based on Bayesian statistics. In this particular loss function, instead of random sampling as done in the BPR algorithm, sampling is performed on negative samples for a particular session and predictions are computed. If the negative item's predictions are greater than the positive item's predictions (rank violation), then a large or small gradient update is made corresponding to the number of samples picked up to compare [3]. We leveraged the implementation by the LightFM library for this method to train the model. We considered different number of latent factors and maintained the number of latent factors at n = 50. Additional parameters such as epochs etc. were kept at 3000 and learning rates were automatically determined through the adagrad learning rate schedule (as it showed superior performed over adadelta).

5.3.Alternating Least Squares(Matrix Factorization) Algorithm

The Alternating Least Squares algorithm proposed by Y.Zhou et al. is another state of the art matrix factorization algorithm that uses the Root Mean Squared Error (RMSE) loss function to identify the user and item matrices that result in the lowest loss in predictions in comparison to the actuals. We leveraged the implementation by the LightFM library for this method to train the model. We considered different number of latent factors and found that the number of latent factors at n = 50 was providing top performance. Additional parameters such as epochs etc. were kept at 3000 and learning rates were automatically determined through the adagrad learning rate schedule (as it showed superior performed over adadelta).

5.4. LightFM (Hybrid Matrix Factorization) Algorithm

The Hybrid collaborative filtering algorithm, LightFM is a state of the art matrix factorization algorithm proposed by M. Kula that brings both content and collaborative filtering together to provide recommendations to the user. As our recommendation requirements corresponds to a lot of new sessions, the goal is that by leveraging features related to the items, we will be able to build better personalized rankings for the in-session prediction. This is typically called as cold-start recommendation problem [13] . In order to model the features, we create item features and item feature categories that are passed to the model as additional features. Internally, the algorithm creates individual matrices for each feature subspace with the user or item representation being a sum across all feature spaces. We finetuned the model with the best model having number of latent factors at n = 50. Additional parameters such as epochs etc. were kept at 3000 and learning rates were automatically determined through the adagrad learning rate schedule (as it showed superior performed over adadelta). Features included item related features (feature_category_id and feature_category_value).

5.5. Multi-Layer Perceptron (Deep Learning) Algorithm

This model looks at the problem as a normalized classification problem. The users and items are fed as one-hot encoded vectors into the embedding layer than converts it into dense vector representations of the item that are then entered into a series of dense layers as part of the MLP stack to predict the normalized score of the particular user-item. This is trained and the model is built according to it. During inference time, the users and items for prediction are provided to the trained model and scores are outputted, thereby allowing to find the top K ranks (100 ranks in our case) from the model.

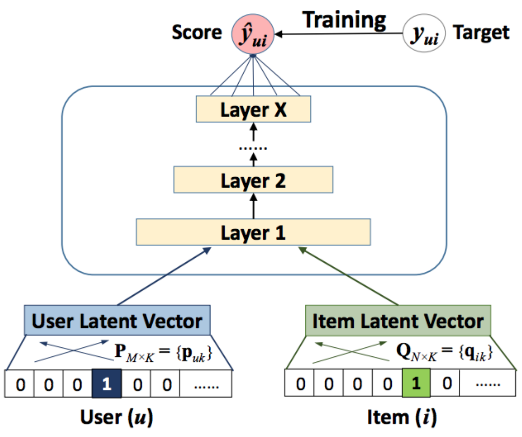

Figure 5 : Multi-Layer Perceptron Model

The finetuned model had 50 dimensions for the output of the embedding layer for users and items. Also, three dense layers with relu activation function were added and a final sigmoid layer that predicts the score of the particular interaction. The actuals were normalized scores between 0 and 1 and were used to train the model via backpropagation. The loss function was cross entropy. The optimizer selected was Adam with learning rate schedules performing learning rate decay to ensure the model doesn't skip the minima. Additionally, dropouts were added after every dense layer to avoid overfitting with percentage of dropouts fixed at 20%.

5.6. Neural Collaborative Filtering (Deep Learning) Algorithm

This model approaches matrix factorization from a deep learning perspective using dense layers and matrix factorization to be done via a deep learning framework. To do this, the embedding layer as per the previous model (MLP) is created and the fed to two models – (1) Generalized Matrix Factorization model that does a dot product of the two vectors to create a similarity vector output and (2) MLP Layer that is similar the one created in the previous modeling approach. The users and items are fed as one-hot encoded vectors into the embedding layer than converts it into dense vector representations of the

[2] https://www.recsyschallenge.com/2022/

item that are MLP stack and GMF stack and finally both are concatenated and fed into a series of dense layers before making a final score prediction using cross-entropy.

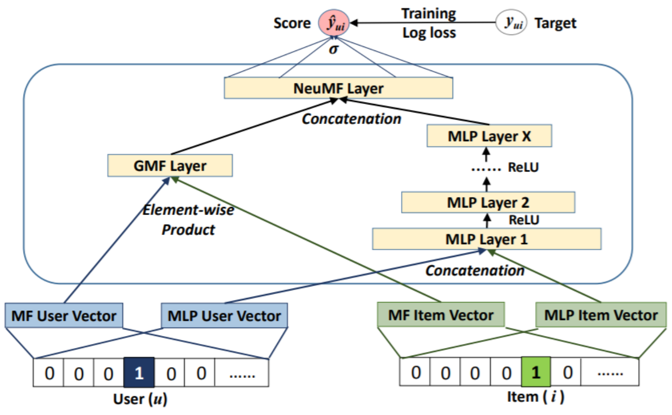

Figure 6 : Neural Collaborative Filtering Model

The finetuned model has 50 dimensions for the output of the embedding layer for users and items in both models. Also, three dense layers were added in the MLP, two dense layers in the NeuMF layer and a final sigmoid layer that predicts the score of the particular interaction. The actuals were normalized scores between 0 and 1 and were used to train the model via backpropagation. The loss function was Cross Entropy. The optimizer selected was Adam with learning rate schedules performing learning rate decay to ensure the model doesn't skip the minima. Additionally, dropouts were added after every dense layer with percentage of dropouts fixed at 20%.

4. EXPERIMENT DESIGN

The experiment was designed with the dataset split into training and validation/test. The training dataset consists of the first 17 months of data (1MM sessions) and the latest 1 month of data (50,000 sessions) is represented in the test dataset. The training data are sessions of users who bought something in a particular session. The way sessions are defined are by looking at all the items viewed in a particular day that have culminated in a Purchase. The session has all item views up to and not including the first view of the item that was bought on that day. The test dataset contains query sessions to generate recommendations for.

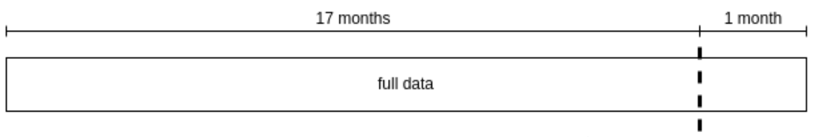

Figure 7 : Train - Test Split

All the metrics generated are provided on evaluation data provided by the challenge organizers. These sessions do not have prior affinities that can be taken advantage of such as user preferences etc. As formulated in the above sections, the goal is to understand how to recommend products for purchase based on in-session data and item features only.

In terms of resources, the experiment was designed to take advantage of both CPU and GPU resources that are available as part of University of Luxembourg's HPC Cluster and free resources such as Kaggle, Google Colab, Sagemaker Studio etc. Both Matrix Factorization methods and Deep Learning methods required HighMemory instances to be available with CPU & GPU compute as sparse matrices of size 1.05MMx23684 needed to be loaded and complex optimization algorithms/matrix factorization algorithms needed to be processed. With respect to the deep learning workloads, the training was performed on GPUs in the HPC cluster and inferences were done in CPU instances owing to limited

GPU availability and high memory requirements, while Matrix Factorization was done completely in CPU instances owing to the nature of the algorithms. In terms of timing, Matrix Factorization methods required ~6 hours to perform each experiment (3000 Epochs with early stopping) and deep learning based methods required ~14 hours per experiment for both training and inference (3000 Epochs with early stopping). The experiments were parallelized as much as possible to reduce overall training and inference times for the experiments.

5. EVALUATION METRICS

Several evaluation metrics such as Area under the curve, Root Mean Squared Error, Mean Reciprocal Rank and precision@k were evaluated as part of the evaluation statistics for assessing performance of the algorithm. However, owing to the problem statement itself and the importance of recommending products with high accuracy with minimal session length, evaluation was reduced to Mean Reciprocal Rank (MRR) as a way to judge the effectiveness of the model. To understand more, The mean reciprocal rank is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer: 1 for first place, 1⁄2 for second place, 1⁄3 for third place and so on. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}$$

Therefore, the Mean Reciprocal Rank is an effective evaluation metric that can optimize higher ranks for recommendations that ultimately were purchases. In our model, we will calculate the MRR for 100 recommendations per session.

6. RESULTS

The results in our paper are geared towards understanding and identifying opportunities to improve the state of the art when it comes to collaborative filtering approaches with respect to Fashion recommendation.

Model Type	*Type*	*Model Name*	*Parameters*	*Epochs*	*Mean Reciprocal Rank (MRR)*	*Recsys Rank*
Matrix Factorization	**Explicit**	BPR	Purchase = 10 ; View = 0	3000	0.008	-
Matrix Factorization	**Explicit**	WARP	Purchase = 10 ; View = 0	3000	0.009	3253
Matrix Factorization	**Explicit**	WARP	Purchase = 100 ; View = 0	3000	0.089	-
Matrix Factorization	**Explicit**	ALS	Purchase = 10 ; View = 0	3000	0.009	3258
Matrix Factorization	**Explicit**	ALS	Purchase = 100 ; View = 0	3000	0.057	2890
Hybrid	**Explicit**	LightFM (Hybrid)	Purchase = 10 ; View = cnt	3000	0.001	3249
Deep Learning	**Explicit**	MLP	Purchase = 10 ; View = cnt	3000	0.034	-
Deep Learning	**Explicit**	NeuCF	Purchase = 10 ; View = cnt	3000	0.021	-

[2] https://www.recsyschallenge.com/2022/

Matrix Factorization	**Implicit**	BPR	Purchase = 10 ; View = cnt	3000	0.081	2756
Matrix Factorization	**Implicit**	WARP	Purchase = 10 ; View = cnt	3000	0.082	2734
Matrix Factorization	**Implicit**	ALS	Purchase = 10 ; View = cnt	3000	0.057	2888
Hybrid	**Implicit**	LightFM (Hybrid)	Purchase = 10 ; View = cnt	3000	0.044	2934
Deep Learning	**Implicit**	Multi-Layer Perceptron	Purchase = 10 ; View = cnt	3000	0.065	-
Deep Learning	**Implicit**	NeuCF	Purchase = 10 ; View = cnt	3000	0.079	-
Matrix Factorization	**Implicit**	BPR	Purchase = 100 ; View = cnt	3000	0.085	-
Matrix Factorization	**Implicit**	WARP	Purchase = 100 ; View = cnt	3000	0.097	-
Matrix Factorization	**Implicit**	ALS	Purchase = 100 ; View = cnt	3000	0.095	-
Hybrid	**Implicit**	LightFM (Hybrid)	Purchase = 100 ; View = cnt	3000	0.043	3002
Deep Learning	**Implicit**	Multi-Layer Perceptron	Purchase = 100 ; View = cnt	3000	0.091	-
Deep Learning	**Implicit**	NeuCF	Purchase = 100 ; View = cnt	3000	0.095	-

Table 1 : Experiment Results

**All of them have early stopping enabled but 3000 is the limit set due to computational power involved.*
+ Progress can be viewed across the weeks in the leaderboard in Appendix along with top scores.
** Missing Recsys ranks are attributed to the closing of the competition submissions Mid-June.*

The three questions I will aim to answer though the paper will be based on the results that are captured as part of the table above -

6.1. IMPLICIT VS EXPLICIT FEEDBACK MODELS

Does modeling the problem as an implicit recommender model outperform explicit recommender systems in the space of Fashion recommendations?

By comparing the model results of the algorithms over implicit and explicit feedback, it is clear that all the models that use implicit feedback outperform the ones in the explicit feedback. This could be the because of the presence of additional information related to the views of the products that are additionally captured within the implicit model that is completely missing from the explicit model because of the scores being whether a purchase has happened or not. This phenomenon is in line with the studies by H.Steck at al. (2015) on the implicit models outperforming explicit models across a wide variety of benchmarks typically. Therefore, a clear recommendation for fashion recommendations is to go with implicit models as they capture more information about the interactions.

6.2. HYBRID VS DEEP LEARNING/VANILLA MATRIX FACTORIZATION

Does adding item level characteristics improve the performance of the models?

In this question, we aim to see whether adding content related to the item feature space improves the model. However, when performing this experiment across both explicit and implicit models, we can clearly see that the mean reciprocal ranks reduce (recommendation quality reduces) significantly. This could be because of the presence of skewed features that are not representative of the item overall such as hierarchies or the likes of them. As a result, the resultant model is not improving the performance in comparison with other state of the art models across deep learning and matrix factorization. Therefore, we can conclude with this approach that the features we have are not good representations of the products or items. We need to either engineer the features to be more representative or obtain further data points to be able to better reflect them. However, with the current state, pure collaborative filtering or deep learning models outperform the hybrid recommendation engines.

6.3. DEEP LEARNING VS STATE OF ART MATRIX FACTORIZATION

Does deep learning architecture beat the performance versus state of the art matrix factorization methodologies.

In this question, we aim to understand whether moving to deep learning in comparison with state of the art models brings improved performance to the model. Overall, from Table 1, we can see that deep learning models outperform several algorithms and are in-line with the better algorithms in the pure collaborative filtering space in the implicit model approach. Post tuning the model, we can see that Neural Collaborative Filtering (NeuCF) algorithm performs almost as good as the WARP (best model) in the matrix factorization domain. This shows that there is architecture of the neural networks may need to be tweaked to be able to look at alternate ways to understand the complexity of the interactions. In this space, there is additional research that can be done by looking at content related features to see if it improves the model (in spite of it not improving using LightFM). Additionally, we can look at sequence to sequence models such as GRUs or RNNs to model the sequence nature of the transactions leading to a purchase.

7. LIMITATIONS AND FUTURE WORK

The current system has two limitations that are areas of research and improvement. Currently, the model is provided sessions across the 2 years for which data has been given to have the same user preferences and trends. In the fashion industry, trends continuously change across the years as clearly outlined by Dressipi as part of the competition rules. The first improvement would be to introduce a decay parameter that reduces the weightage of older transactions in comparison to newer transactions or simply reduce the training data to include only the recent transactions. This will allow the model to index more towards newer transactions while predicting item recommendations.

Another area of improvement is the way cold start is handled. Currently, sessions not seen by the deep learning model are calculated using the weighted average of nearest sessions from the nearest neighbor algorithm. This is already a major challenge in current recommendation engines and is clearly outlined by M.Kula in his paper "Metadata embeddings for user and item cold-start recommendations". We can improve the way we introduce features in the deep learning framework by introducing some learnings from the work done by Covington et al. in the paper "Deep Neural Networks for YouTube Recommendations". We can create additional embedding layers based on item features and concatenate them before the MLP layers. This will allow to build more complex hybrid approaches using Deep Learning approaches. Additionally, more extensive feature engineering can be done to tune the model performance even further based on category clusters etc.

8. REFERENCES

[1] Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009.

[2] Weston, Jason, Samy Bengio, and Nicolas Usunier. "Wsabie: Scaling up to large vocabulary image annotation." IJCAI. Vol. 11. 2011.

[3] Weston, Jason, Hector Yee, and Ron J. Weiss. "Learning to rank recommendations with the k-order statistic loss." Proceedings of the 7th ACM conference on Recommender systems. ACM, 2013.

[4] Zeiler, Matthew D. "ADADELTA: An adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[2] https://www.recsyschallenge.com/2022/

[5] Duchi, John, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” The Journal of Machine Learning Research 12 (2011): 2121-2159.

[6] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

[7] Zhou, Yunhong, et al. "Large-scale parallel collaborative filtering for the netflix prize." International conference on algorithmic applications in management. Springer, Berlin, Heidelberg, 2008.

[8] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." 2008 Eighth IEEE international conference on data mining. Ieee, 2008.

[9] He, Xiangnan, et al. "Neural collaborative filtering." Proceedings of the 26th international conference on world wide web. 2017.

[10] Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." Proceedings of the 10th ACM conference on recommender systems. 2016.

[11] Kalantidis Y, Kennedy L, Li L (2013) Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In: International conference on multimedia retrieval, pp105–112

[12] Nogueira K, Veloso AA, dos Santos JA (2016) Pointwise and pairwise clothing annotation: combiningfeatures from social media. Multimed Tools Appl 75(7):4083–4113

[13] Kula, Maciej. "Metadata embeddings for user and item cold-start recommendations." *arXiv preprint arXiv:1507.08439* (2015).

[14] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003, doi: 10.1109/MIC.2003.1167344.

[15] Zhang, Aston, et al. "Dive into deep learning." arXiv preprint arXiv:2106.11342 (2021).