The Movie Recommendation Design for Millennials

Chen Yu Wang
c.wang4@students.uu.nl

Introduction

We propose a NN embedding based collaborative filtering (CF) framework, namely, [1]embedding matrix factorization, which can integrate any kind of side information effectively to make diverse movie recommendations with minimum churn rate. We generate latent factors by the feature transforming function direct from users and movies ratings instead of estimating it from the observed data. The model will transform explicit data based on the user's rating to a low-dimensional key patterns vectors, and also permit possible solutions for future improvement expansions by adding more implicit features like age, the region in the evaluation matrix to achieve better performance.

1 Purpose of Design

1.1 Persona

The word **"Millennials"** (**Generation Y**), generically refers to people born in the 1980s and 1990s, a time when people were in the midst of a technological boom (Wikipedia, Millennials). Generation Y is often receptive to new things and brave enough to accept challenges compared to Generation X, which is conservative and pessimistic. Their attitude towards life can be viewed as mostly self-centered, flexible, and timely. But sometimes it is very easy to make the value bias while receiving a rapid message flow.

1.2 Problem: Echo chambers

Because of the advancement in technology, human interactions and values are based on opinion-based communities, where person only encounters information or opinions that reflect and reinforce their own. The phenomenon also has a professional term, **Echo chambers**, which means the person only hears the same voice again and again.

1.3 metrics, values and stakeholders

As mentioned earlier, our stakeholders: Millennials, it's easy to accept new things and embrace new technologies. Therefore, how to judge the right values becomes very important. Unlike the design for the X generation users, focusing on how to let older feel cozy when watching the video with a large component. The recommended design for millennials should focus on how to **diversify the recommended content** and consider the level of **Churn rate loss** while increasing diversity.

[1] Code is available at https://github.com/JimCurryWang/ADS_recommendation

For instance, if we only recommend a preferable movie, we can certainly maintain the Churn rate very well. Nonetheless, once we try to diversify to recommend new content to the user, we don't know whether the user may like it or not. If users don't like the content, there is great potential that they lose interest, become less adherent, and eventually leave the platform. While it remains very important to diversify the film's content. **Diversity** helps us to disseminate the multicultural values to our users, helping them develop the critical ability to think logically (R.Hu & P.Pu, 2011). More importantly, diversity can break of the boundaries of **Echo chambers**, allowing users to hear different voices and avoid overly radical conceptual tendencies.

2. Algorithm

Recommender systems have two primary approaches: collaborative filtering or content filtering. **Collaborative filtering** relies on the concept that people who liked something in the past would also like a similar experience in the future. **Content-based filtering** is useful where information is known about the item but not about the user. It models a classifier to evaluate the likes and dislikes of the user concerning the characteristics of an item.

In this article, we will focus primarily on collaborative filtering approaches, and three models are used to construct our film recommendation system. (1) **USER-USER Similarity Matrix** (2) **ITEM-ITEM Similarity Matrix** (3) **Embedding Matrix Factorization (EMF)**.

2.1 USER-USER and ITEM-ITEM Similarity Matrix

USER-USER: Convert to the most similar user group, view their frequently watch films and recommend them to current target user.

ITEM-ITEM: Identify the movie groups that are most similar to the films you are currently browsing and recommend them to your users.

There are numerous ways of calculating similarity: Jaccard similarity, Pearson similarity, and Cosine similarity. In here we use pearson similarity, however, regardless of which type of concordance filtering we use, we treat each column in the matrix as a vector and calculate its to obtain the similarity value.

Generally, if there are many potential users, the USER-USER Similarity Matrix will be very large and time-consuming to compute, so it is faster to use ITEM-ITEM collaborative filtering. Instead, in the case of a content-oriented recommendation, such as **new film released**, the USER-USER method might be more suitable to use. Additionally, the USER-USER method recommends the most-watched movies, so it often used to recommend **pøpular movies**, while the ITEM-ITEM method is easier to recommend **long-tail movies** (not much-watched, but continuously watched). The two types of recommendation methods are believed to have a positive effect on the overall churn rate and diversity (see memory_cf.ipynb).

2.2 Embedding Matrix Factorization

In 2.1, we use memory-based collaborative filtering, which requires calculating the similarity of each cell in a table. This exhaustive method will consume a lot of computing resources. Hence, we try to propose a neural network based collaborative filtering framework, namely, **Embedding Matrix Factorization(EMF),** which can integrate any kind of side information effectively and also mitigate the sparse matrix problem. (see emf-nn.ipynb[2])

2.2.1 Rating Matrix

Suppose $V \in R^{n \times m}$ is the rating matrix of m objects and n users, then each row V_i represents all ratings of a person, V_j represents all people's ratings for a certain movie, and V_{ij} represents ratings of a movie j by a person i. For corresponding movie recommendations, V must be a sparse matrix and has many null values because the number of movies is huge (Ma, 2008). (Exactly, the problem we have in 2.1)

However, we can use two matrices U and M to mimic the rating matrix as V.

$$V=UM^T$$

U is the matrix of user preference and M is the matrix of movie features. Each row of $U \in R^{f \times n}$ denotes a user and each column denotes a feature. The values indicate the correlation between a user and a feature, and the larger the value, the more distinct the feature. Each row of the matrix $M \in R^{f \times m}$, represents a movie and each column represents the association of the movie with a feature.

In fact, there is a diagonal matrix S in the middle of the right-hand side of the equation, which we can take as multiplying and merging with U (The S matrix is left blended into the feature vectors, so only U and M remain). If we want to predict whether the user u likes the movie i, one can use the inner product.

Algorithm 2.2 Embedding Matrix Factorization

Select a learning rate μ, batch size z, hidden feature dimension d
1. Set the starting values of matrices U, M.
2. Define the embedding dimension for hidden feature and values matrices U, M.
3. Repeat
    (a) Compute gradients descent on ∇U and ∇M with batch size = z.
    (b) Set $U \leftarrow U - \mu\nabla U$, $M \leftarrow M - \mu\nabla M$.
  until the validation MSE loss goes into converge
END

[2] Code is available at https://github.com/JimCurryWang/ADS_recommendation/blob/main/emf-nn.ipynb

2.2.2 Adjustment term

Typical collaborative filtering data exhibits large systematic tendencies for some users to give relevant lower or higher ratings than others. Therefore, we consider adding some biased terms to the rating process based on personal taste and the level of the film.

$$r_{ui} = u + b_i + b_u + q_i^T p_u$$

u = Constant denoting the over all average rating
b_i = bias for item i
b_u = bias for user u
q_i = bias latent factor
P_u = user latent factor

The term b_u indicates the objective factors in the user's rating habits that are not related to the item(film). For example, some users are more demanding and have high expectations and requirements in a movie, then their ratings will be lower, while some users are more tolerant and feel good about everything, then their ratings are generally higher. The same reasoning for the adjustment like items b_i and the global adjustment u. (Figure 2)

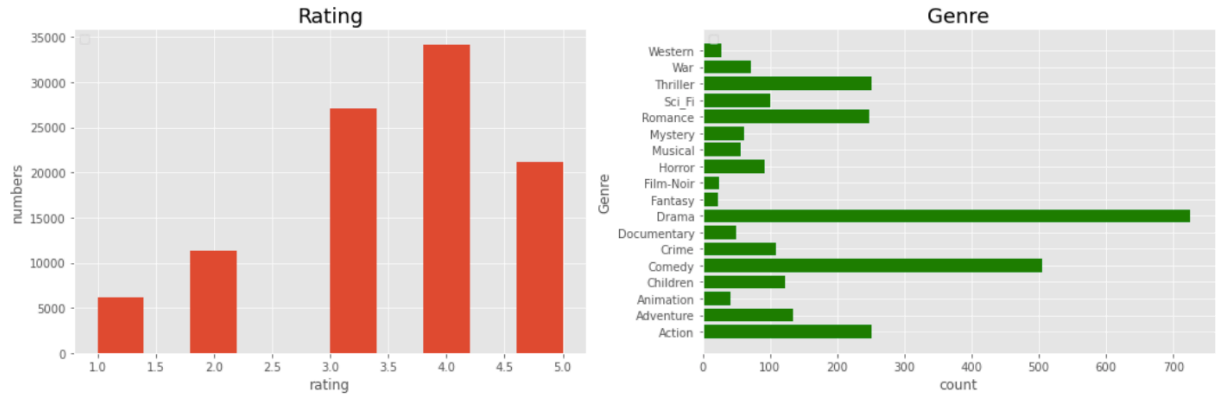

Figure 1 Explore Data Analysis by user's rating and movie types. We can find that most of the user generally give 3-4 rating, however, there are also some movies got rather low rating by 1-2, which should be consider in the adjustment term in our model. (left) The movie genre type concentrate on Drama and Comedy (right) (see EDA.ipynb[3])

3 Result

3.1 application

In our attempt to make the calculation as less time-consuming as possible, the **EMF model** work well. Unlike the SVD by doing the matrix factorization, we try to use nn.Embedding as another ways to reach the similar goal. Another perk in using this kind of structure is that it can easily be added on to for instance with other explicit data like Television Content Rating System (IMBD), published region and Director, or implicit data like watching times and traffic path. Some data

[3] Code is available at https://github.com/JimCurryWang/ADS_recommendation/blob/main/EDA.ipynb

can be collected by **web scraper** (see scraper.ipynb) and put into our embedding matrix then through backpropagation and gradient descent to find the minimal loss.

In the usage scenario with the interface, the USER-USER matrix can be used in **hot movies recommended**, whereas the ITEM-ITEM matrix can be applied in **latest released movie** or classically **memorable movie** because of the long-tail feature. As for our most powerful model, EMF-net allows users to discover films that match their tastes based on the past preference. Therefore, can be applied when a large number of movies are presented in the same time and need to control the presented numbers like movie list in **category page**, the order of the search results optimize from the **search bar**, and the content of the **homepage banner** recommends pairings with other conditions adjustments (see Appendix).

```python
class EMF_Net(nn.Module):
    '''embedding user and items from explicit rating records
       ref : Matrix Factorization Techniques for Recommender Systems
       ref : Probabilistic Matrix Factorization

       Parameter:
       ----
       len_u : number of users
       len_v : number of items
       dim : feature dimensions of lantent factor u and v
    '''
    def __init__(self,len_u,len_v,dim):
        super(EMF_Net,self).__init__()
        self.U = nn.Embedding(len_u,dim)
        self.V = nn.Embedding(len_v,dim)
        self.Ub = nn.Embedding(len_u,1)
        self.Vb = nn.Embedding(len_v,1)

    def forward(self,user, item):
        u_lant = self.U(user-1)
        v_lant = self.V(item-1)
        u_bias = self.Ub(user-1).squeeze_()
        v_bias = self.Vb(item-1).squeeze_()
        score = u_lant.view(1, -1).mm(v_lant.view(-1, 1)).squeeze_()
        score = score + u_bias + v_bias + mu
        return score
```

Figure 2 Python Code for EMF model with adjustment term (see emf-nn.ipynb)

```
User 196
from rating, he/she like
movie_id  predict_rating  movie_title
=========================================================
285            5       Secrets & Lies (1996)
663            5       Being There (1979)
153            5       Fish Called Wanda, A (1988)
286            5       English Patient, The (1996)
692            5       American President, The (1995)


from rating, he/she might like
movie_id  predict_rating  movie_title
=========================================================
64            4.45       Shawshank Redemption, The (1994)
50            4.42       Star Wars (1977)
12            4.41       Usual Suspects, The (1995)
174            4.36       Raiders of the Lost Ark (1981)
408            4.32       Close Shave, A (1995)
```

Figure 3 EMF Net output result. i.g. userid = 169

3.2 performance

Our solution EMF-net, reach a **MSE loss** around 0.91 and **accuracy score** around 0.72.

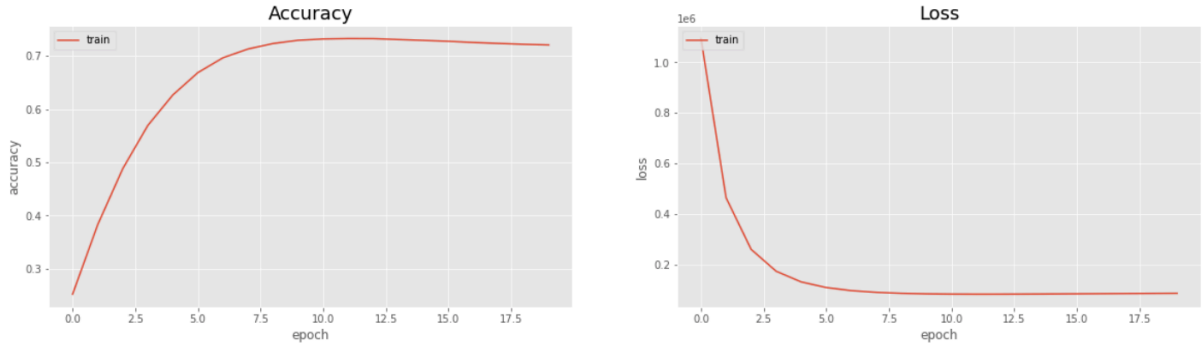

Figure 4 The Accuracy rate and MSE loss of EMF Net (see emf-nn.ipynb)

4 Conclusion

We present a more modularity design to movie recommendation with NN embedding based collaborative filtering, which can integrate other side information in the future effectively to make diverse movie recommendations with minimal churn rate. The usage scenario can be referred to our prototype website https://www.circlecircle.net/, and also see in the Appendix. In the interface design, we use circle shape instead of rectangle one to make user feel more intimacy while browsing, concise interface to make visual more intuition and **algorithmic affordances** design like Incognito (Figure 9). Last, our model can expansion by others explicit data like what we do with scraper to make recommendation more diversity, but this need further improve with some unsolved problems like "**Cold Start**" and "**latent feature explode**".

(See more code: https://github.com/JimCurryWang/ADS_recommendation)

5 Reference

Chih-Chao Ma, A Guide to Singular Value Decomposition for Collaborative Filtering (Ma, 2008) https://www.csie.ntu.edu.tw/~r95007/thesis/svdnetflix/report/report.pdf

Wikipedia, Millennials (Wikipedia, Millennials)
https://en.wikipedia.org/wiki/Millennials

IMBD, Television content rating system (IMBD, Television content rating system)
https://help.imdb.com/article/contribution/titles/certificates/GU757M8ZJ9ZPXB39#

Rong Hu and Pearl Pu, Helping users perceive recommendation diversity. (R.Hu & P.Pu, 2011)
http://ceur-ws.org/Vol-816/divers2011.pdf#page=53

MovieLens
https://grouplens.org/datasets/movielens/

Algorithmic Affordances, Koen van Turnhout (Turnhout, 2021)
https://aapatternlibrary.wordpress.com/

6. Appendix

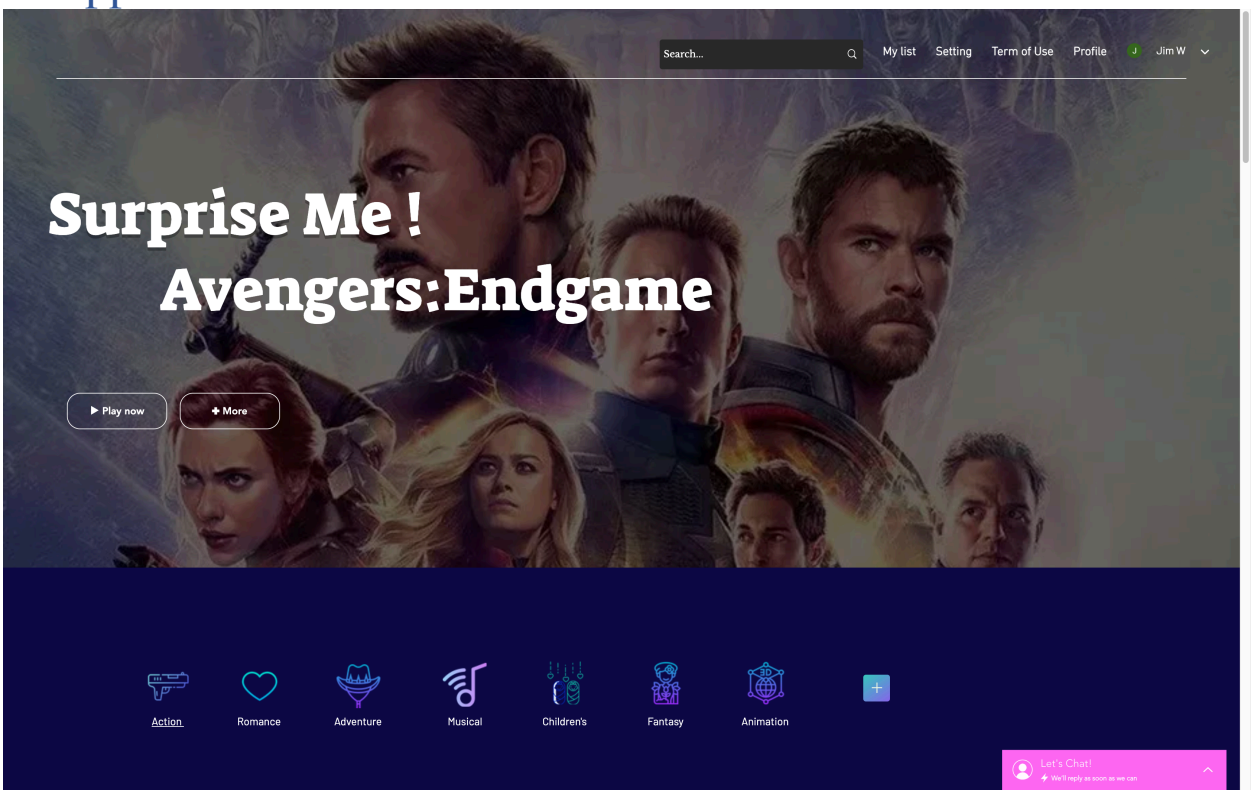

Figure 5 home page with EMF-Net.

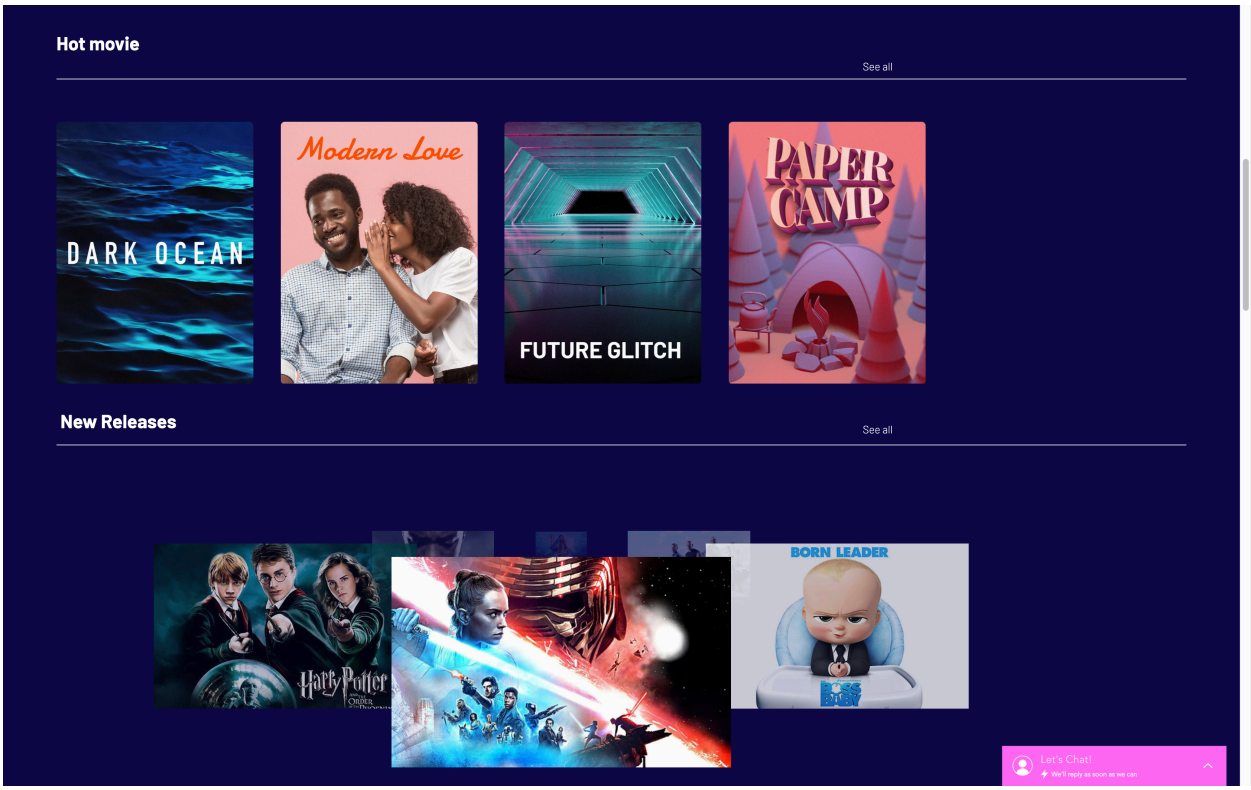

Figure 6 Hot movie with User-User CF and New Releases with Item-Item CF because of its rating feature.

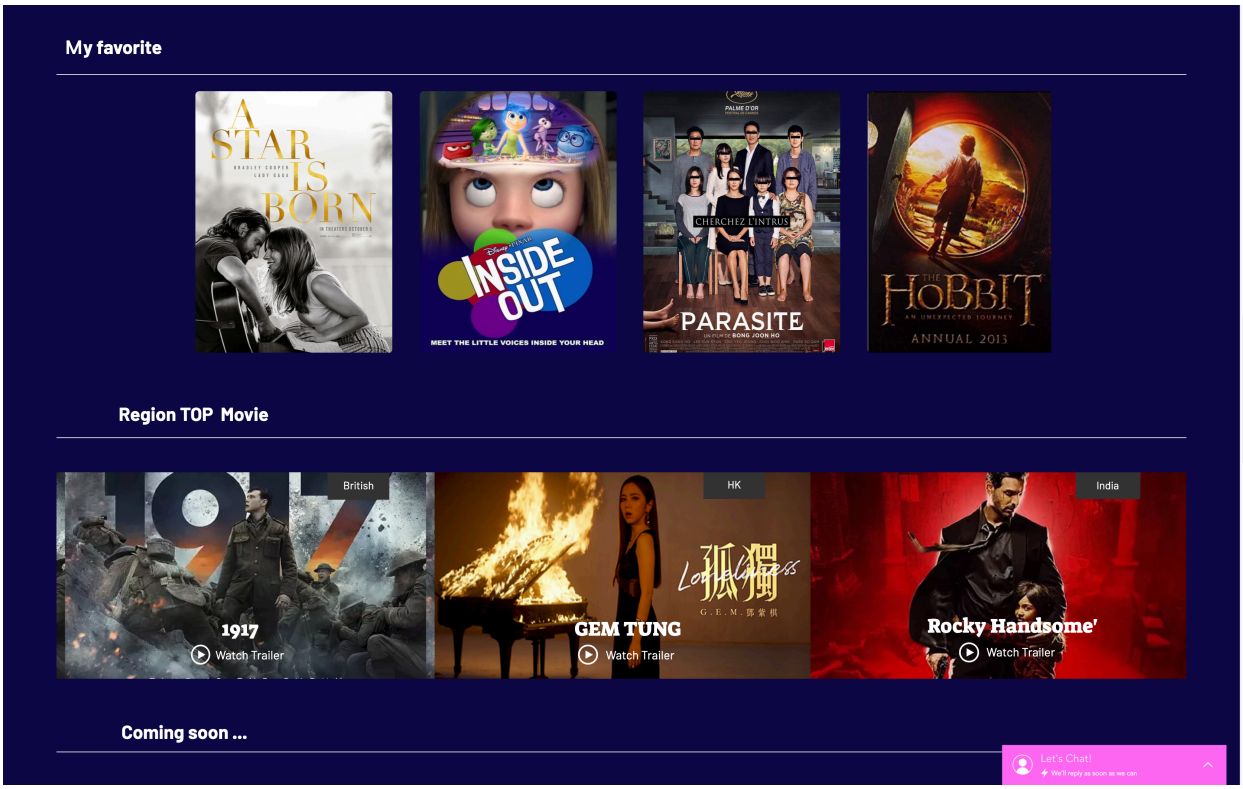

Figure 7 Diversify the recommendation with different culture content. i.g. Hong Kong, Korean, USA movies.

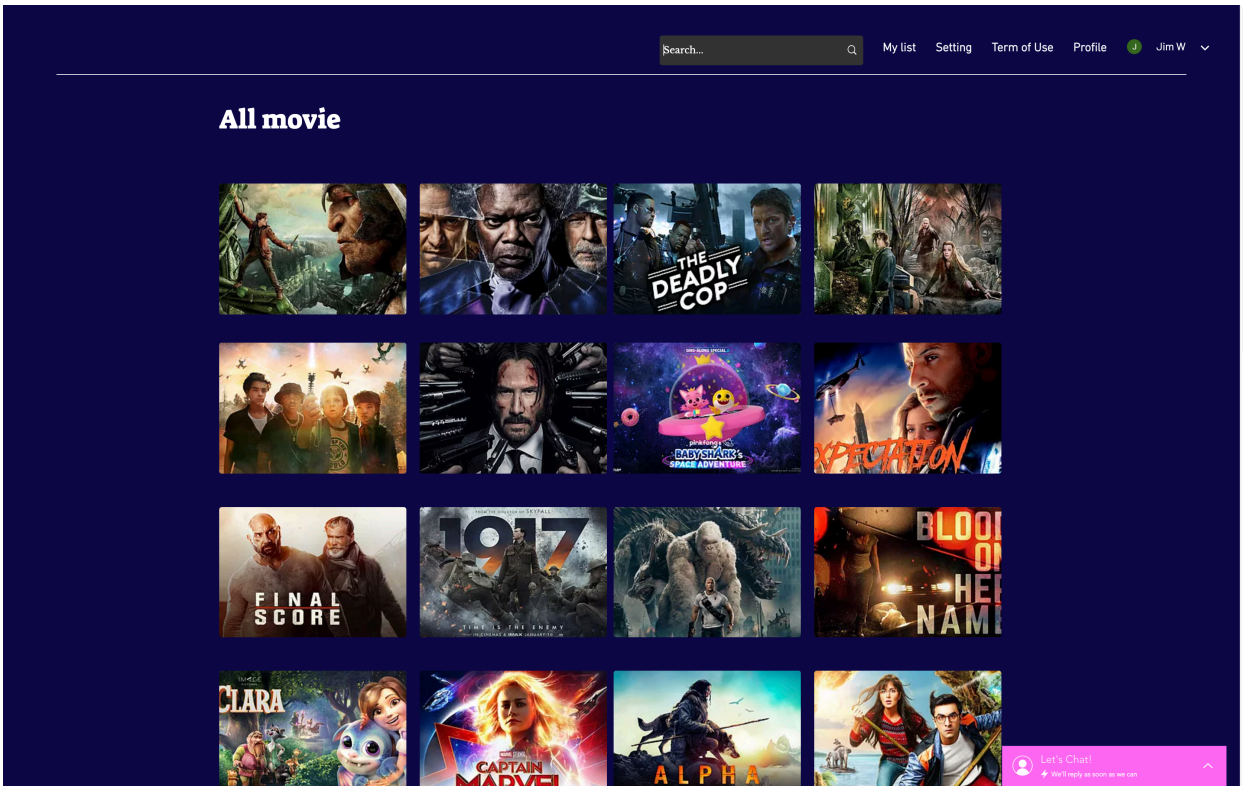

Figure 8 Show the most interesting movies to user via EMF-Net with preference order when content pop out in the same times.

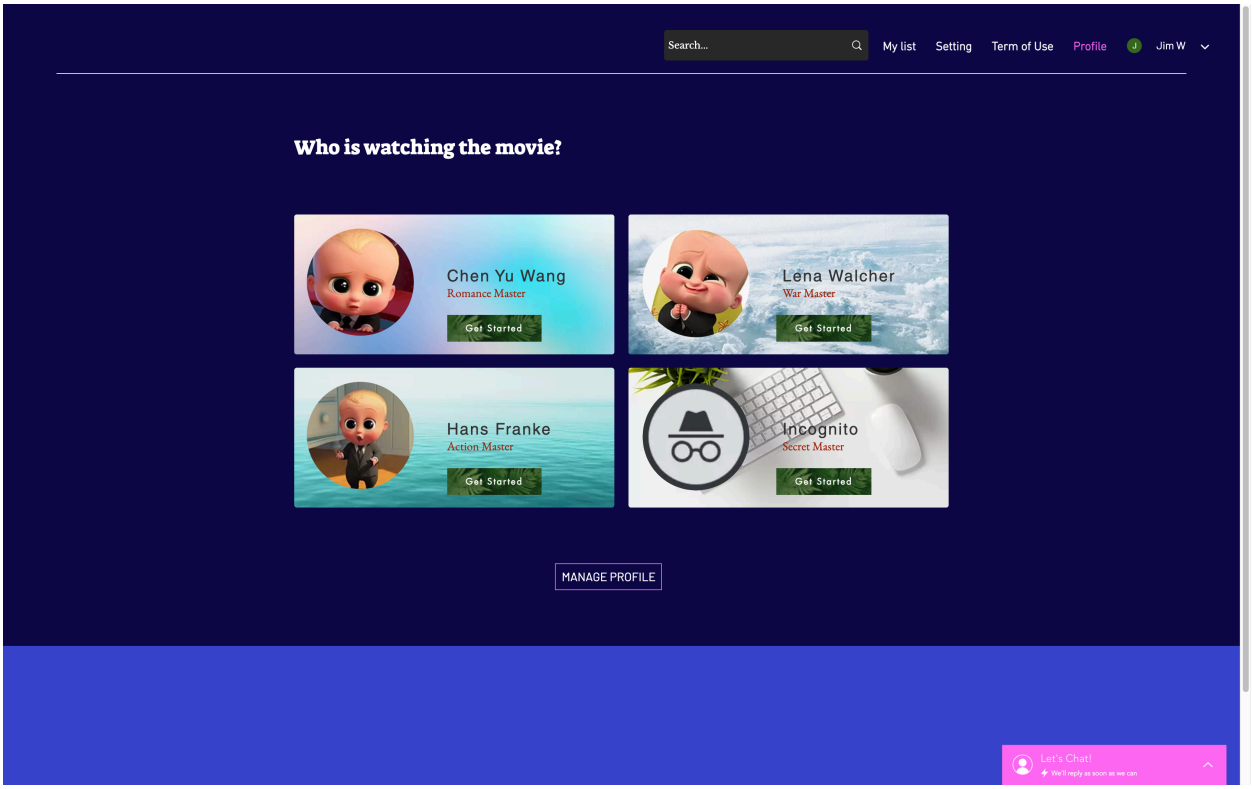

Figure 9 Algorithmic Affordances design: Multiple Profiles - Sharing account with your family/friends in one subscription. Incognito - User can be enabled to watch things without the algorithm learning from it with an incognito functionality. (Turnhout, 2021)