UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Movie recommender systems: a study on evaluation, and perspective for neural networks usage.

Author:
Yevheniia MITRIAKHINA

Supervisor:
Hanna PYLIEVA

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Yevheniia MITRIAKHINA, declare that this thesis titled, "Movie recommender systems: a study on evaluation, and perspective for neural networks usage." and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"The whole of life is just like watching a film. Only it's as though you always get in ten minutes after the big picture has started, and no-one will tell you the plot, so you have to work it out all yourself from the clues."

Terry Pratchett, Moving Pictures

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Movie recommender systems: a study on evaluation, and perspective for neural networks usage.

by Yevheniia MITRIAKHINA

Abstract

Recommender systems are designed to assist in information search whenever information is abundant. Classic examples of a situation where choices need to be made without sufficient personal experience are found in e-commerce, entertainment (audio and video content), and social media platforms. The focus of this work is on a classical application of recommendation systems to movie recommendations and experiment are conducted on real-world data provided by Ukrainian online cinema platform. In this project we implemented neural and classic collaborative filtering techniques. Our experiments show that neural collaborative filtering significantly improves the performance of a recommender system. Also, it is proposed to augment collaborative filtering techniques with content-based approach to improve diversity, coverage, and novelty of the system.

Acknowledgements

I would like to thank my supervisor, Hanna Pylieva for her encouragement, useful and timely suggestions.
I am grateful to Serhii Ivanets (sweet.tv) for providing the data for the project and giving me insights into how the big-scale systems are structured.
Thanks to my family and all of my friends for their never-ending support and encouragement.
And last but not least, thanks to Ukrainian Catholic University for providing me with a scholarship, introducing to a truly amazing community and making this four-year journey challenging but truly delightful.
Honorable mentions to all of the ice cream for supporting my mental health in the days of completing this thesis.

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

1 Introduction **1**
- 1.1 Brief History 1
- 1.2 Related works 2

2 Experimental setup **4**
- 2.1 Data overview 4
  - 2.1.1 Preprocessing 4
- 2.2 System design 6
- 2.3 Evaluation 8
  - 2.3.1 Test Split 8
  - 2.3.2 Functional metrics 8
  - 2.3.3 Non-functional metrics 9

3 Experiment outcomes **11**
- 3.1 Content-based approach 11
  - 3.1.1 Strategy design 11
  - 3.1.2 Evaluation of the content-based model 13
- 3.2 Collaborative filtering 14
  - 3.2.1 Generalized matrix factorization 14
  - 3.2.2 Multi-layer perceptron 15
  - 3.2.3 NeuMF 16
  - 3.2.4 Evaluation of results 17

4 Conclusions **19**
- 4.1 Contribution 19
- 4.2 Future work 19

Bibliography **20**

List of Figures

3.1 Strategy pattern used for structuring content-based recommender. 12
3.2 GMF architecture 15
3.3 MLP architecture. 16
3.4 NeuMf architecture 17

List of Tables

2.1 Sparsity of the dataset before and after sampling. 5

3.1 Non-functional metrics for content-based system. 13
3.2 Non-functional evaluation of collaborative-based approach . . 18
3.3 Functional evaluation of collaborative-based approach 18

List of Abbreviations

HR	**H**it **R**atio
IMDB	**I**nternational **M**ovie **D**atabase
MAP	**M**ean **A**verage **P**recision
MF	**M**atrix **F**actorization
MLP	**M**ulti **L**yaer **P**erceptron
NDCG	**N**ormalized **D**iscounted **C**umulative **G**ain
NeuMF	**Ne**ural **M**atrix **F**actorization
TF-IDF	**T**erm **F**requency **I**nverse **D**ocument **F**requency

To discipline - the highest of virtues that I wish to possess

Chapter 1

Introduction

A recommendation system is defined as an information filtering system that provides a decision-making strategy for users under complex information environments via narrowing and prioritizing the items (Rashid et al., 2002). They are an integral part of e-commerce websites, audio and video content platforms, and news platforms. It has been estimated that 35% of items purchased on Amazon and 75% of shows watched on Netflix come from product recommendations (Meyer and Noble, 2013).

1.1 Brief History

Although it can be argued that the first recommendation systems were library catalogs and tour guides, we focus on automated recommendation systems that emerged in the 1990s with an increased amount of information being available. One of the first examples of such systems was Tapestry, developed by Xerox in 1993 (*History of recommender systems*). It allowed filtering a large stream of electronic documents into a smaller stream, satisfying users' personal interests, and prioritizing selected documents. It employed both collaborative and content-based strategies (Terry, 1993).

Research efforts in recommender systems and online communities were first summarized within the GroupLens research project at the University of Minnesota started in 1992. They developed a collaborative recommendation system for Usenet - a discussion service that emerged in the early days of the Internet that was operating via a system of newsgroups. Simultaneously, thematic recommenders were emerging, like Ringo - a music recommendation service developed by MIT (*History of recommender systems*). Now GroupLens maintains the MovieLens dataset and research project that has grown over the years and reached a stable benchmark of 25 million ratings and many other projects concerning spatial, social, and crowdsourced data.

The realization of the commercial value followed shortly. One of the pioneers in the field was Amazon that presented its research paper on Item-to-Item Collaborative Filtering in 2001 ("The history of Amazon's recommendation algorithm. Collaborative filtering and beyond"). Amazon's system shifted the focus from user to item perspective, meaning that recommendation was based on the visitor's recent purchase history and, for each purchase, drew a list of related items. It achieved a new degree of personalization relative to other collaborative approaches, allowing to base recommendation on short-term activity and had computational benefits as a single user could only buy a tiny fraction of items present on the website.

The new wave of research and excitement concerning recommendation systems followed with the announcement of the Netflix Prize in 2006 that asked to outperform the company's CineMatch system by 10%(*Netflix Prize* 2006). It triggered the development of more efficient matrix factorization techniques. Also, the idea of stacking algorithms together and combining their outputs gained attention to the point where the winning solution ended up consisting of 107 different algorithms.

At the current state of advancement, the importance of proper understanding of the data, precise formulation of value proposition, and dependence on the business domain evaluation metrics were recognized by the research community. Looking forward, recommendation systems are a dynamic field where the researchers and industrial practitioners are still trying to figure out the issues connected to a temporal, contextual recommendation, scale, and effective recommendation is still a craft (Konstan and Ekstrand, 2017).

1.2 Related works

The recommendation system's objective can be generally classified into two types - prediction and recommendation (Aggarwal, 2016). The main difference between them is that prediction aims to guess the rating for a particular user-item pair. In contrast, recommendation aims to select a subset of items that the user will like or a subset of users who will like the item and order them.

Recommender systems can operate on content-based information - in the case of movies that can be a description, director, genre, as well as on implicit and explicit user feedback. Explicit user feedback is obtained when the user reviewed the item upon request, for example - liked or disliked, or put a mark from one to five. Implicit user feedback is collected from the activity of the user, such as clicks, search queries, viewing time. Implicit data is usually more abundant and provides honest insights into user's preferences because of response biases in explicit rating. User-based information such as age, location, and gender can also be utilized, although they are not available for some applications.

Collaborative and content-based are the central strategies of analyzing data in the context of recommendation (Aggarwal, 2016). Generally, the collaborative approach centers around finding a neighborhood of users with similar interests and, for each user - finding and recommending what other similar users have reviewed positively. Collaborative filtering algorithms face a number of challenges including sparsity of the data, natural scarcity of negative feedbacks, scaling for new users and new products (Su and Khoshgoftaar., 2009). Collaborative-based systems do not include item or user features and operate solely on interaction data.

In contrast, the content-based approach aims to learn the representation of the items that the user rated via content- or user-related rich features and propose to the user items whose features are the closest to such representation according to the similarity metric of choice, for example cosine similarity, Pearson correlation or simply Euclidean distance (Pazzani and Billsus, 2007). Content-based approaches completely ignore interaction data.

With a recent increased interest in machine learning algorithms, some of them gave the beginning to new strategies in recommendation.
Sequential methods aim to view the user's interaction with items as a time-dependent series of events rather than an unordered collection of events and mainly utilize self-attention and recurrent networks. Sequential methods are considered effective for session-based prediction - whenever a user is likely to interact with multiple items throughout the session (Hidasi et al., 2016b). They mainly utilize recurrent, long short term memory and gated recurrent units(Tan, Xu, and Liu, 2016).

Knowledge-based methods formulate recommendations in terms of knowledge graph search problem (Cao et al., 2019). Building the knowledge graph is a data-intensive task and such systems are in the early stages of development so they are out of scope of this work.

Context-aware approaches aim to give the recommendation given contextual information such as time, location, company, etc., and use a wide array of algorithms from factorization machines (Rendle, 2010) to deep learning (He and Chua, 2017).

Neural networks are used not only to incorporate entirely new strategies but became somewhat of an industry standard. Youtube, which has one of the largest scale-wise recommender systems in existence, declares that two neural networks are used within the service. One is intended for candidate generation and operates in a collaborative filtering paradigm, and the other is used for ranking (Covington, Adams, and Sargin, 2016). Neural networks are also applied for feature extraction from audio, video, and text (Deldjoo et al., 2018, Deldjoo, 2018, Hidasi et al., 2016a).

All things considered, recommender systems are a very diverse field, and the consensus on the optimal algorithm or its evaluation is not yet reached.

Chapter 2

Experimental setup

2.1 Data overview

The data for this project was kindly provided by sweet.tv[1] - Ukrainian online streaming platform that provides subscription-based access to a variety movies, TV-series and TV-channels.

Data spans ten months of user ratings from about 2000 users who watched about 4000 unique movies. Content-related information about 4400 movies is available. Content information includes title, year of production, IMDB rating, text description, genres, director, actors, writers, music composer, art director, and producer.

The only data available about the user concerns views of the particular item and the time when the view occurred. Optionally, comments are present. As there is not enough data in the comments and they are provided in different languages (Ukrainian and Russian), comments are not considered further. There are approximately 213 thousand views available. The platform does not collect demographic or other person-identifiable data.
Ratings are not available in neither explicit nor implicit form i.e., likes or dislikes or the session duration.

2.1.1 Preprocessing

Following preprocessing was applied to the movies content-based data:

1. Calculation of popularity score done on the whole train set based on the number of views on the platform.
2. Calculation of proxy "quality" score for actors, writers, music, art, and producers. The score is developed with the idea that introducing all agents(i.e., actors or writers) will dramatically increase the dimensionality of the data. On the other hand, relative popularity score helps differentiate less popular agents from more popular ones. It suggests

[1]https://sweet.tv/

that popularity scores of similar movies are similar, thus making it possible to calculate the similarity of the agents with metrics such as cosine similarity.

3. One-hot encoding of the most popular genres.
4. Preprocessing of text descriptions: stopwords filtering, lemmatization, stemming, sentiment score calculation.

The views data as a typical example of implicit data is open for interpretation. As only the fact of a view is available, and not its duration or relative duration, we are presented with a challenge of identifying positive and negative samples. Having a distinction between positive and negative samples is essential for collaborative filtering models used later. The assumptions for generating samples from views are:

1. The majority of views are coming from movies posted on the front page of the platform in collections such as 'popular movies' or 'premieres.'
2. Only a tiny portion of the movies (about 7%) belongs to collections.

Therefore, if a person has not viewed a movie from collections, we can state with a great degree of confidence that they have seen the information for the movie but remained indifferent to it. There is also a possibility that the user has watched a movie before joining the platform but in such case, labeling it a negative sample is also a valid option as we do not want it showing again in the recommendations.

As a result, the views present in the dataset receive a synthetic score of 5, and 20 random movies from collections that the user has not watched receive a score of 1. The choice of a number of negative samples is somewhat arbitrary, based on the minimum sought number of samples for a user. It is not larger as then it might amplify known issues of the approach including additional noise to the model.

Sparsity of the dataset calculated as

$$Sparsity = \frac{\text{number of interactions}}{\text{number of users} \times \text{number of items}}$$

Dataset	Sparsity
Original user-item matrix	95.6%
Augmented with negative samples matrix	94.1%

TABLE 2.1: Sparsity of the dataset before and after sampling.

Side effect of negative sampling is that sparsity is reduced and is now comparable to benchmark's MovieLens-1m dataset (93.6%).

2.2 System design

To refine the requirements for the recommendation system, we use a framework proposed in (Konstan and Ekstrand, 2017) whose eight dimensions of analysis are briefly outlined below:

- **Domain** refers to what specifically is an output of RS, e.g., individual item, sequence, or bundle of items. It also includes whether the recommended item is new or has been seen/bought by the user before. Domain restricts a set of items subject to business rules such as showing only items that are in stock, have not been disliked by the user, relevant to the group of users with similar tastes.
- **Purpose** refers to a broader business vision for RS, such as to drive sales on certain high-margin items, gain competitive advantage via providing additional value for the user, improve quantitative indicators of engagement, provide education about system's features, or assist in building a community.
- **Recommendation context** describes what the user is doing when the recommendation is displayed and the form in which it is proposed, be it the next song that will be played automatically after the previous one or ten products in the section "frequently bought along with item X."
- **Whose opinions are considered?** Broadly speaking, all RS are based on some opinion. The opinion may come from experts, an entire set of users, a neighborhood of people who have similar taste or have picked an item before, or previous user activity.
- **Personalization level** can be standard - everyone sees the same thing, e.g., most popular items; tailored to demographic characteristics of the user; based on the user's short-term activity and long-term history of interests.
- **Privacy and trustworthiness** define the amount of control and transparency the user has over their data used and the recommendations provided.
- **Interfaces** are characterized as the way that the output is presented. For example, is it a prediction of rating or a recommendation of a selection of items? Are they presented in an organic way similar to search results, or the fact that it is a recommendation is explicit, and the user can interact with it?
- **Algorithms**, obviously, are the drivers of the RS.

We propose a recommender system composed of three distinct components. The reason for having separate components in a recommender system is to diversify the predictions range. Diversification is indeed needed because the service does not possess enough data about user's movie preferences before they joined the platform. Also, the assumption is that movie interests are rarely homogenous, e.g. if a person has been viewing dramas lately it does not mean they will not view a comedy in another setting.

Analysis for each of them is provided below:

"Movies similar to X" / "if you like this movie, you will also like"

Domain	All available movies that the user did not watch.
Purpose	Provide value for the user via having a better selection of movies. Demonstrate possibilities for the premium package.
Context	View with details of movie X is augmented with a panel displaying a selection of similar movies of dynamic size depending on popularity of the movie.
Opinions	A set of users who had watched movie X and a minimum of other 10 movies on the platform.
Personalization	User's current session activity is considered. Long-term activity is not used. A set of users whose opinions are utilised for the recommendation is chosen purely on the basis of activity on the platform, no demographic or otherwise personal data is used.
Privacy	The user cannot evaluate or influence the recommendation.
Algorithms	Content-based filtering.

"You may like others from genre N"

Domain	All available movies not watched by the user that belong to the selected genre.
Purpose	Provide value for the user via reducing time needed to find an item of interest and increasing the chances to find suitable content. For the product - increase user satisfaction and user consumption leading to lesser churn.
Context	Homepage for the user has a tab with the most recent genres they watched and five picks for each genre.
Opinions	A set of users who had watched movies in genre N and rated them positively or neutrally.
Personalization	User's long-term activity, previous implicit and explicit feedback related to the genres and movies within the genre.
Privacy	Users can implicitly evaluate the recommendation via adding to the "watch later" list, viewing the film immediately; or explicitly choosing "not interested" for the genre or particular movie.
Algorithms	Content-based filtering.

"Top picks for you"

Domain	All available movies not watched by the user
Purpose	Provide value for the user via reducing time needed to find an item of interest and increasing the chances to find suitable content. For the product - increase user satisfaction and user consumption leading to lesser churn.
Context	Homepage has an additional view with top ten movies that is personalized to each user.
Opinions	User's long term activity, similar users long term activity
Personalization	User's long-term activity, previous implicit and explicit feedback related to movies.
Privacy	Users can implicitly evaluate the recommendation via adding to the "watch later" list, viewing the film immediately; or explicitly choosing "not interested" for the particular movie.
Algorithms	Experimentally determined among: collaborative filtering, NeuMF.

2.3 Evaluation

2.3.1 Test Split

We use the leave-one-out cross-validation method to evaluate the model, commonly applied to the recommender system evaluation (Pu, Chen, and Hu, 2011). Out of the sequence of user's views, the last item is chosen as a test set and the one before last as a validation set. As a result of the execution of any of the proposed models, we get ten items. Both test and validation samples are compared with these ten items.

2.3.2 Functional metrics

Accuracy metrics focus on evaluating whether a predicted rank matches the actual rank provided by the user. Classical accuracy metrics are mean absolute error, and root mean squared error. They are indeed useful for training a machine learning model but are of little use for the problem of filtering and ranking items.

Decision support metrics include precision, recall and f1-score. They focus on what fraction of recommendations are suitable (or unsuitable) for the user. While it is important that they somewhat satisfy an objective for RS, they fail to consider rank and cover the whole dataset. Thus, the inaccurate recommendation in top-5 items is punished the same way as it is in the 90-100 position.

Rank-aware metrics include mean reciprocal rank, mean average precision(MAP), normalized discounted cumulative gain(NDCG) and hit ratio. Rank-aware metrics, unlike accuracy and decision-support, allow to evaluate both the relevance of the item and the rank that it was assigned. We calculate hit ratio as

$$HR = \frac{\text{number of sets of recommendations that contain a relevant item}}{\text{number of all sets of recommendations}}$$

A set of recommendations consists of 10 movies. Relevant item is defined by the last movie that the user have chosen (the one that belongs to the test set). MAP@10 and NDCG in our case are defined as

$$MAP@10 = \frac{1}{index(\text{relavant item}) + 1}$$

$$NDCG = \frac{log(2)}{log(index(\text{relavant item}) + 2)}$$

Both metrics are set to 0 if the relevant item is not present in the recommendations set. For calculation of the metrics on the whole dataset we take mean of all individual values.

For training models we use a combination of hit ratio, MAP and NDCG.

2.3.3 Non-functional metrics

Accuracy metrics provide a relatively narrow view of the performance of a recommender system. Moreover, test data is inherently biased because it is based on historical user choices that may be affected by the system's current design and which items it ranked high. Due to the limited insights provided by accuracy-based metrics, researchers introduced different formal and informal quality measures. These measures commonly include diversity, stability, adaptivity, novelty, serendipity, and coverage (Pu, Chen, and Hu, 2011, Ge, Delgado-Battenfeld, and Jannach, 2010).

Quality defines to which degree the system is meeting or exceeding customer's expectations. Quality is a highly informal measure that can be roughly approximated by business metrics such as click- and view-through rate, one-day or test-period retention, and subscription renewal rates.
Coverage concerns the degree to which recommendations cover the set of available items and the extent to which the system can generate suggestions to all potential users.
Novelty estimates the degree to which recommendations are original, non-trivial, and not apparent to the user.
Serendipity is concerned with the both relevance and novelty of recommendations and how far recommendations may positively surprise users.
Diversity measures how different the recommended items are based on, i.e., the number of categories they belong to or the distance between the items' vector embeddings. Diversifying can also be introduced as a measure to

boost the consumption of items from the "long tail" of popularity distribution (less popular ones).
Stability evaluates the consistency of recommendation algorithms when it is presented with the new data.

We attempt to measure and report some of the non-functional quality metrics as well as functional to create a more complex view on the system's capacity.

Chapter 3

Experiment outcomes

3.1 Content-based approach

An advantage of using a content-based approach is that there is no need for extensive data from other users as a recommendation is specific to the user. Considering that in the case of movies, the majority of items have comprehensive descriptions and other features that allow diversifying the vector representation of each item, a content-based approach is a valid choice. The cold start issue is reduced compared to collaborative filtering to having at least one movie rated positively by the user.

One of the limitations of this approach is that the model builds recommendations based on the user's present interests and has a limited ability to extrapolate outside of the limited space that the user has already seen. Also, the content available for analysis is limited and, to a large extent, hand-engineered. Descriptions that do not provide specific information about the item, i.e., the plot or emotional characteristics, use varying vocabulary and style present a unique challenge for automatic interpretation.

3.1.1 Strategy design

In our implementation, a content-based strategy consists of two subtasks: filtering and ranking. Filtering enables to choose a subset of all movies of size n % from the initial dataset that minimizes the cosine distance from the target movie. It is recommended to stack multiple filtering strategies to achieve better results. Available filtering strategies include:

TF-IDF Relevance Filter. When applied to movie descriptions, the TF-IDF filter evaluates how the weighted most important terms from the original description are similar to those in other movie descriptions via calculating term frequency-inverse document frequency score. Denote by $tf(t, d)$ frequency of term t in a document d and by

$$idf(t, D) = log \frac{\text{number of documents}}{\text{number of documents where term t appears}}$$

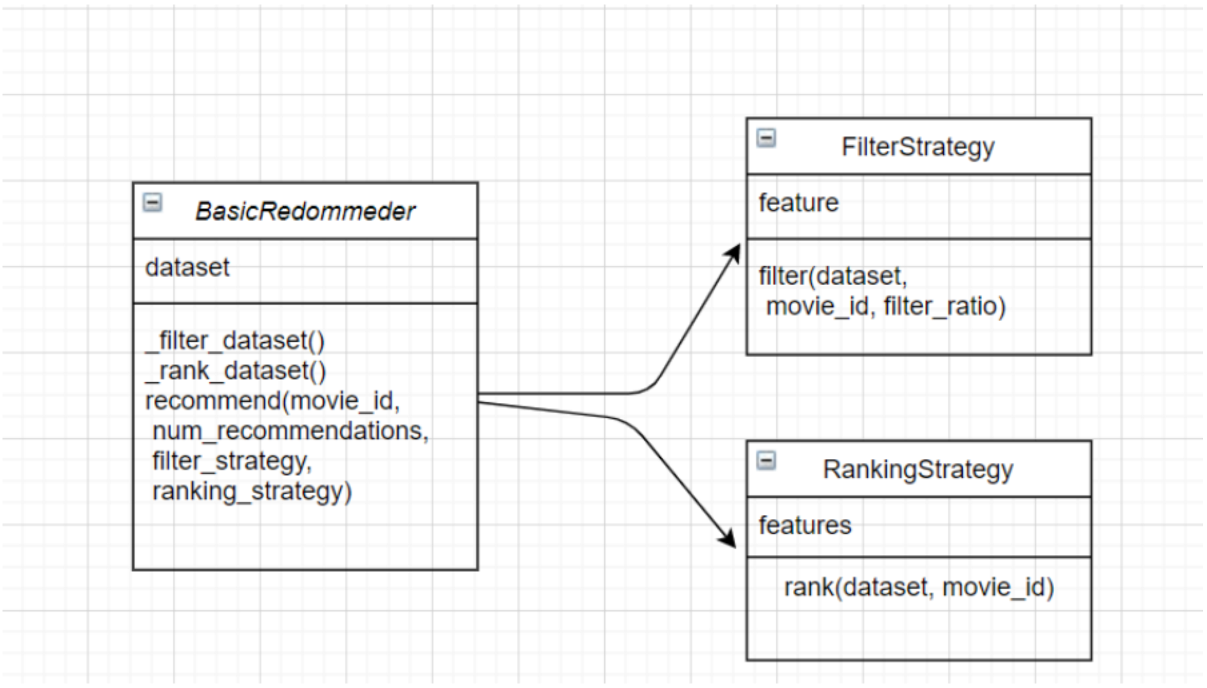

FIGURE 3.1: Strategy pattern used for structuring content-based recommender.

Then TF-IDF score is calculated as

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

TF-IDF provides a weighted measure of relevance of a search query composed of movie description in other descriptions. From the observations - this approach yields not obvious yet intuitively similar movie descriptions. For example, when looking for similar movies for a sentimental biopic, the top results will include biographies. The problem in this particular example is that biographies can be very different in sentiment and genre. That is why we augment with the following filters.

Genre Similarity Filter. As movies most often belong to several genres, the score to compare them is calculated as follows:

$$similarity = \frac{|\text{target genres} \setminus \text{other movie genres}|}{|\text{target genres}|} + \frac{|\text{other movie genres} \setminus \text{target genres}|}{|\text{other movie genres}|}$$

where $set1 \setminus set2$ is a set difference operation and $|set|$ is a set size.

This approach works best on the data that has been pre-filtered as the space of genres is too little to filter out most of the movies. However, it helps to restrain the diversity introduced by the TF-IDF filter, thus improving relevance.

Agent Similarity Filter. Agent similarity is calculated based on the cosine similarity of the popularity scores introduced during the preprocessing stage. The idea behind it is to help differentiate between, for example, a Hollywood movie with top actors and art-house movies that will inevitably have lower scores in an attempt to draw a line between these different styles of production.

The **ranking** is another essential subtask in the content-based strategy. It is universally recognized that the order of the recommended items is as important as the selection of items. Most evaluation metrics for recommendations are weighted by rank in one way or another. The ranking is mainly based on numerical features such as IMDB score, the popularity score, or the year of production and can be applied in highest to lowest (for a single feature) or cosine similarity manners.

3.1.2 Evaluation of the content-based model

The primary purpose of the content-based engines is to introduce additional diversity and provide better coverage for the items from the "long-tail" of items popularity distribution. That's why non-functional metrics are more informative of the approach's quality.

To estimate the quality of the system we construct a test dataset that contains 10 content-based predictions based on penultimate historical view for each user. We reserve the last view to use as a test set. On this dataset we estimate coverage, diversity, novelty and hit ratio.

Metric	Value
Coverage, users	99.47%
Coverage, movies	91.9%
Movies, recommended less than 50 times (novelty)	97.5%

TABLE 3.1: Non-functional metrics for content-based system.

The hit ratio for the approach reaches only 1.4%. However, as the algorithm could generate predictions for the vast majority of users, and most movies have been recommended at least once, we can state that it satisfies the criteria for coverage. Also, most movies are recommended less than 50 times which signals about the recommendations' diversity and novelty.

3.2 Collaborative filtering

We adopt a collaborative filtering approach to utilize the information about user interactions, which can contain valuable insights that cannot be described by regular item features. Unlike in many e-commerce applications, the approach is user-based since it is required by the system's design.

We test classic matrix factorization vs. neural collaborative filtering introduced by He et al., 2017. As the data is implicit, the user-item matrix construction process undergoes some modifications. Let m be the number of users and n - number of items. Then interactions matrix $R \in \mathbb{R}^{m \times n}$ is constructed as:

$$r_{ui} = \begin{cases} 1, & \text{if the interaction is observed} \\ 0, & \text{if interaction is not observed} \end{cases}$$

3.2.1 Generalized matrix factorization

Matrix factorization creates an embedding for each user and item that is that entity's representation in the latent space. The latent space is simply a representation of compressed data in which similar data points are closer together in space.

Matrix factorization creates an embedding for each user and item that is the entity's representation in the latent space. The latent space is a representation of data in which similar data points are close in space. Matrix factorization evaluates rating r_{ui} for user u and item i as an inner product. Let u_n be the vector embedding for some user and i_m - for an item. Then estimated rating $\widehat{r_{ui}}$ is given by

$$\widehat{r_{ui}} = u_n^T i_m$$

MF is a linear model, which means it can capture only linear dependencies in the data. Linearity limits the expressiveness of the model and is a considerable limitation. The attempts to overcome these limitations include introducing a user bias term and increasing the number of dimensions in the latent space which can lead to poor generalization of the model.

In our implementation generalised matrix factorization (GMF) is used. GMF is a network that aims to approximate the factorization of matrix R as a product of matrices of user's latent features and item's latent features. It uses stochastic gradient descent to minimize binary cross-entropy of differences between actual and predicted r_{ui}. Binary cross-entropy is given by

$$h = -\frac{1}{n} \sum_{u=0,i=0}^{n,m} r_{ui} * log(p(r_{ui} + (1 - r_{ui}) * log(1 - p(r_{ui}))$$

where $p(t_u i)$ is an output probability of rating being 1. Binary cross-entropy helps to reflect binary nature of implicit feedback during training. Adam optimizer is used for stochastic gradient descent.

The inputs to the GMF model are column u for user embedding and row i for item embedding that we take from matrix R and pass to an embedding layer. GMF is implemented using Keras[1] framework and resulting model is composed as follows:

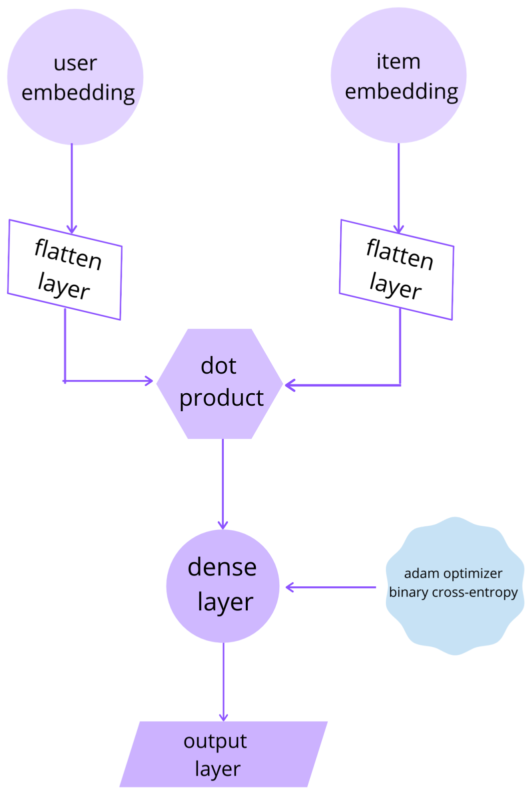

FIGURE 3.2: GMF architecture

As a result we obtain a model with 222,831 trainable parameters (number of missing values in the interaction matrix). The GMF model only needs one epoch to train and than it stops improving as the model is simply linear.

3.2.2 Multi-layer perceptron

Within a neural collaborative filtering framework, matrix factorization is combined with a deep neural network that is supposed to introduce non-linearity into the model. This way, we take advantage of both linearity of MF and the non-linearity of a neural network that feasibly extends the expressiveness of the model via introducing additional complexity.

[1]https://keras.io/

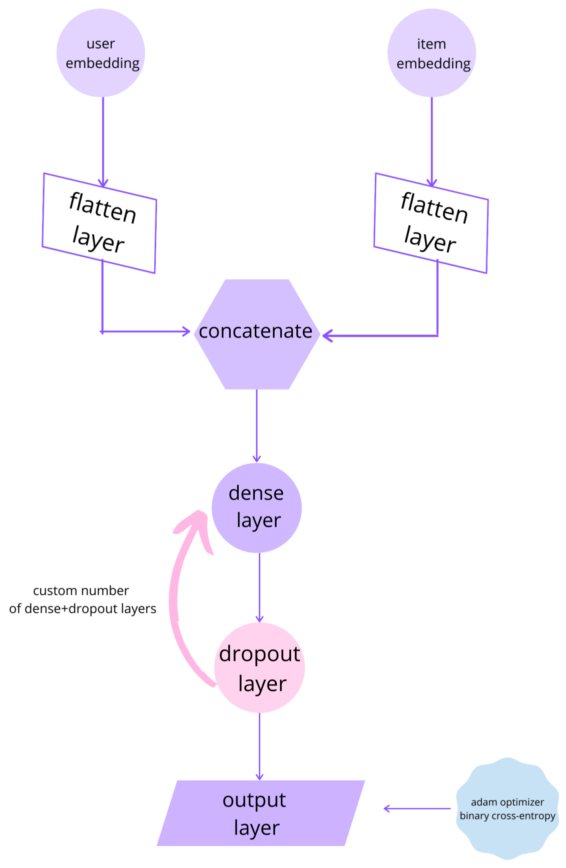

FIGURE 3.3: MLP architecture.

We follow the approach introduced in the paper (He et al., 2017) and choose multi-layer perceptron (MLP) as a neural component. MLP is one of the simpler neural architectures that, in our implementation, is composed of three densely connected layers with dropout after each dense layer to combat overfitting. The rate of dropout is 10%. The architecture of MLP is depicted below. The inputs to the MLP model are column u for user embedding and row i for item embedding that we take from matrix R. ReLu is the activation function for the dense layers.

$$ReLu(x) = max(0, x)$$

While the amount of dense layers in the model is easily adjustable, the optimal number of layers, that was experimentally obtained, is between 2 and 5. We use 3 dense layers in the resulting model.

3.2.3 NeuMF

To combine GMF and MLP under the NeuCF framework, we create separate embeddings for matrix factorization and neural network part, resulting in four different embeddings - item MF, user MF, item MLP, user MLP. Model with shared embeddings will likely not exploit the benefits of both algorithms.

Then, the corresponding embeddings pass through GMF and MLP, and the results are combined within the last hidden layer with sigmoid activation.

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

The network is trained to minimize the logloss between actual and predicted rating and uses adam [2] optimizer. The model stops significantly improving after 15 epochs of training, so we select the best iteration within 15 epochs to predict.

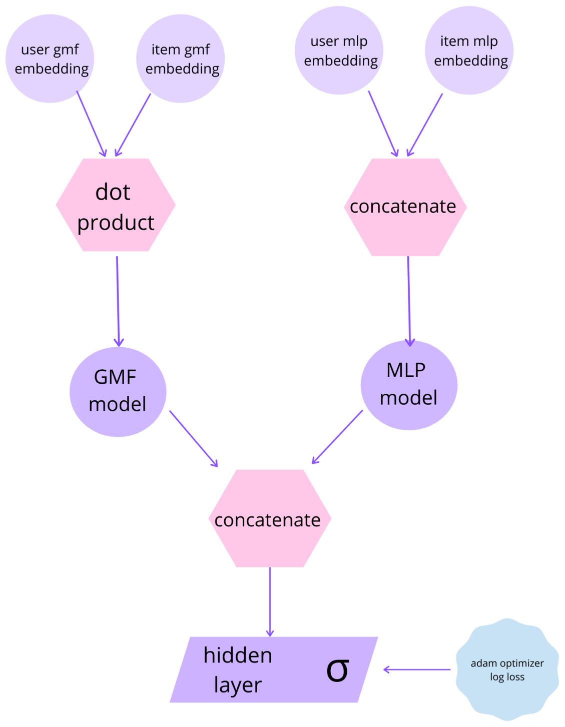

FIGURE 3.4: NeuMf architecture

Considering that ranking all the missing items for each user is too time-consuming, we randomly select 100 items that the user has not interacted with and rank them. Then ten items with the highest rank are selected, and they constitute our recommendation set. The recommendation set is evaluated for the last and penultimate items.

3.2.4 Evaluation of results

From non-functional metrics follows that NeuMF is less diverse and less novel. It can be explained via the specificity of the algorithm: it achieves better results in emulating the historical behaviour (see table below) and, as

[2] https://keras.io/api/optimizers/adam/

Algorithm	GMF	NeuMF
Coverage, movies	99.03%	55%
Coverage, users	100%	100%
Movies, recommended less than 50 times	83.9%	35.4%

TABLE 3.2: Non-functional evaluation of collaborative-based approach

the users are mostly watching a set of popular movies, the diversity of the algorithm decreases, following the historical behaviour.
Both algorithms achieve complete user coverage.

algorithm	validation			test		
	HR	MAP@10	NDCG	HR	MAP@10	NDCG
GMF	0.405	0.092	0.224	0.403	0.088	0.222
NeuMF	0.519	0.137	0.304	0.502	0.126	0.288

TABLE 3.3: Functional evaluation of collaborative-based approach

As we can observe, NeuCF significantly outperforms GMF (by 11% of hit ratio) in both scenarios. It must be noted that the time needed to train one epoch on both networks is comparable (about 70 seconds), although only one epoch is required for GMF, in contrast to 15 needed for NeuMF.

It must be noted that the NeuMF has a potential for improvement *within* the current architecture we did not tune a number of hyperparameters, including the dimension of users and items latent spaces, regularization parameters, dropout rates, number of neurons in fully-connected layers, learning rates and type of optimizer.

Still, NeuMF outperforms classical matrix factorization-based collaborative filtering by a margin that could win an analogy of Netflix prize. Our results confirm the findings of He et al., 2017 on a new use case and suggest that introducing non-linear kernels to a recommender system is beneficial.

Chapter 4

Conclusions

4.1 Contribution

In this work we considered a problem of movies recommendation on a real-world use-case.

- We studied different ways of assessing the recommender system's quality and implemented the ones that are quantifiable.
- Proposed a recommendation system consisting of three components
- implemented two different recommendation approaches that can be accessed on GitHub[1]
- reported content-based and collaborative filtering's functional and non-functional metrics

This work can be used as a proof-of concept to build a production-scale recommender system based on the data preprocessing, modelling and assessment techniques introduced.

4.2 Future work

We have several directions for future work:

1. Now, neither model is adjusted to cold start issues, i.e. we are not able to predict items for a new user. Collaborative-filtering based system will not be able to predict newly added items. Content-based system can cover new items if they possess a comprehensive description.
2. Testing on historical data does not give a full view on the quality of the model. AB-testing would be required to measure the business-related metrics.
3. Inclusion of non-latent user and item features could provide a valuable addition to the embeddings used currently.
4. There is a potential to adjusting the NeuMF model's hyperparameters.

[1]https://github.com/mitryahina/movies-recommender

Bibliography

Aggarwal, Charu C. (2016). *Recommender Systems: The Textbook*. 1st. Springer Publishing Company, Incorporated. ISBN: 3319296574.

Apáthy, Sándor. *History of recommender systems*. https://onespire.hu/sap-news-en/history-of-recommender-systems/.

Cao, Yixin et al. (2019). "Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences". In: *CoRR* abs/1902.06236. arXiv: 1902.06236. URL: http://arxiv.org/abs/1902.06236.

Covington, Paul, Jay Adams, and Emre Sargin (2016). "Deep Neural Networks for YouTube Recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys '16. Boston, Massachusetts, USA: Association for Computing Machinery, 191–198. ISBN: 9781450340359. DOI: 10.1145/2959100.2959190. URL: https://doi.org/10.1145/2959100.2959190.

Deldjoo Y., Elahi M. Cremonesi-P. et al (2018). "Content-Based Video Recommendation System Based on Stylistic Visual Features". In:

Deldjoo, Yashar et al. (2018). "Audio-visual encoding of multimedia content for enhancing movie recommendations". In: 455–459.

Ge, Mouzhi, Carla Delgado-Battenfeld, and Dietmar Jannach (2010). "Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. Barcelona, Spain: Association for Computing Machinery, 257–260. ISBN: 9781605589060. DOI: 10.1145/1864708.1864761. URL: https://doi.org/10.1145/1864708.1864761.

Hardesty, Larry. "The history of Amazon's recommendation algorithm. Collaborative filtering and beyond". In:

He, Xiangnan and Tat-Seng Chua (2017). "Neural Factorization Machines for Sparse Predictive Analytics". In: *CoRR* abs/1708.05027. arXiv: 1708.05027. URL: http://arxiv.org/abs/1708.05027.

He, Xiangnan et al. (2017). "Neural Collaborative Filtering". In: *CoRR* abs/1708.05031. arXiv: 1708.05031. URL: http://arxiv.org/abs/1708.05031.

Hidasi, Balázs et al. (2016a). "Parallel Recurrent Neural Network Architectures for Feature-Rich Session-Based Recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys '16. Boston, Massachusetts, USA: Association for Computing Machinery, 241–248. ISBN: 9781450340359. DOI: 10.1145/2959100.2959167. URL: https://doi.org/10.1145/2959100.2959167.

Hidasi, Balázs et al. (2016b). *Session-based Recommendations with Recurrent Neural Networks*. arXiv: 1511.06939 [cs.LG].

Konstan, Joseph A and Michael D. Ekstrand (2017). *Introduction to Recommender Systems: Non-Personalized and Content-Based*. https://www.coursera.org/learn/recommender-systems-introduction. Accessed: 2021–03-20.

Meyer, I. and S Noble (2013). *How retailers can keep up with consumers.* Tech. rep. MacKenzie.

Netflix Prize (2006). https://www.netflixprize.com/rules.html. Accessed: 2021-05-01.

Pazzani, M.J. and D. Billsus (2007). *Content-Based Recommendation Systems*. https://doi.org/10.1007/978-3-540-72079-9_10.

Pu, Pearl, Li Chen, and Rong Hu (2011). "A User-Centric Evaluation Framework for Recommender Systems". In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. Chicago, Illinois, USA: Association for Computing Machinery, 157–164. ISBN: 9781450306836. DOI: 10.1145/2043932.2043962. URL: https://doi.org/10.1145/2043932.2043962.

Rashid, Al Mamunur et al. (Jan. 2002). "Getting to know you: Learning new user preferences in recommender systems". English (US). In: 2002 International Conference on intelligent User Interfaces (IUI 02) ; Conference date: 13-01-2002 Through 16-01-2002, pp. 127–134.

Rendle, Steffen (2010). "Factorization Machines". In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. USA: IEEE Computer Society, 995–1000. ISBN: 9780769542560. DOI: 10.1109/ICDM.2010.127. URL: https://doi.org/10.1109/ICDM.2010.127.

Su, Xiaoyuan and Taghi M. Khoshgoftaar. (2009). *A Survey of Collaborative Filtering Techniques*. https://downloads.hindawi.com/archive/2009/421425.pdf.

Tan, Yong Kiam, Xinxing Xu, and Yong Liu (2016). *Improved Recurrent Neural Networks for Session-based Recommendations*. arXiv: 1606.08117 [cs.LG].

Terry, Douglas B. (1993). "A Tour through Tapestry". In: *Proceedings of the Conference on Organizational Computing Systems*. COCS '93. Milpitas, California, USA: Association for Computing Machinery, 21–30. ISBN: 0897916271. DOI: 10.1145/168555.168558. URL: https://doi.org/10.1145/168555.168558.