A HYBRID LATENT REPRESENTATION RECOMMENDER SYSTEM FOR LIVE STREAMING CHANNELS

LAUTARO NICOLÁS PACELLA

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

WORD COUNT: 9090

STUDENT NUMBER

2059102

COMMITTEE

Msc. Paris Mavromoustakos Blom
Dr. Grzegorz Chrupała

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

June 24, 2022

Abstract

The rapid growth of the live streaming phenomenon has brought to platforms like Twitch the challenge of developing systems that can make effective recommendations regarding their available content to improve the user experience. Some research has been done in this field, but only a limited number of approaches have been implemented. This study proposes to analyze the performance of a hybrid recommendation system based on latent representations, using as input a type of interaction specific to live streaming, and channels' metadata. The results show the positive effect of adding side information and hyperparameter optimization on the proposed model, and that it outperforms other simpler approaches.

1 INTRODUCTION

This chapter starts with an introduction to the main topics of the paper, followed by a description of the Twitch.tv platform particularities. Afterwards, it presents the scientific and practical motivation of this work and the research questions it will address. Lastly, it describes the outline of the following chapters.

1.1 *Context*

Live Streaming consists of transmitting digital media content in real-time. Just like in other media transmission resources, the broadcasted content can take various forms, but one of the most consumed genres is video games, with an audience estimated to reach 920 million by 2024 (Newzoo, 2021). At present, twitch.tv is the leading live streaming provider, with 1.3 trillion minutes of content watched, an average of 31 million daily visitors, and more than 8 million unique streamers going live every month, according to their internal data reported on their website (Twitch.tv, 2022). Having such a large amount of content available can be seen as a positive element of the platform, but research has shown that having too many choices can be counterproductive because the expectations and the amount of information to process grow in relation to the options available (Oulasvirta, Hukkinen, & Schwartz, 2009). This suggests the need for the platform to make effective recommendations that can help users find relevant channels.

To this end, businesses and researchers have been developing and improving what we now know as recommender systems (RS). These are tools that can use the data of the items they intend to recommend and the target users to create notions of affinity between them and generate meaningful and useful recommendations (Melville & Sindhwani, 2010). Two of the classic and most used types of RS are Collaborative Filtering

(CF), where the recommendations are based on the decisions that other users with similar preferences have made, and content-based (CB) which makes recommendations based on the similarity of the characteristics of the items chosen by the target user. However, both types of RS have their own disadvantages. CF underperforms on items and users with none or scarce past interactions (Isinkaye, Folajimi, & Ojokoh, 2015). CB recommender systems, on the other hand, are usually very limited by the availability of the content to be analyzed, and they can lead to overspecialized results with a lack of novelty in the recommendations (Khusro, Ali, & Ullah, 2016).

Nevertheless, researchers have also explored the option of combining these methods into hybrid models to reduce their respective disadvantages and bolster their advantages (Çano & Morisio, 2017). These combinations can be done in many different ways, but one specific method proposed by Kula (2015) and showing very promising results, consists of jointly modeling the interactions data and content data into a latent space.

This hybrid RS applies the same principles of matrix factorization methods (see Section 3.3.1) but instead of doing so directly with items and users, it uses the sum of their constitutive features as the input to be factorized into latent representations. This presents some advantages, such as fewer parameters that need to be estimated and better estimations for cold start items and users. Results of experiments with this model show that the hybrid latent representation approach performs at least as well as CF when dealing with dense data, and as a CB recommender system when dealing with very sparse data, and outperforms both when applied to datasets of varying sparsity.

1.2 *Twitch.tv*

Twitch.tv is the leading live streaming service provider, focusing mainly on video games, but extending to other types of content such as daily life events, music streaming, online betting, etc. Currently, they have a RS in place inside the platform, but the specific details about how these recommendations are made have not been disclosed.

Aside from users being passive viewers of content, this platform provides other types of interactions that can be exploited in order to make channel recommendations, for example, the "followings". Following a channel is a free feature of the platform that allows users to keep track of the activity and updates about such channels. It also grants a preferential status to this channel, given that the design of Twitch.tv platform has a dedicated place for channels that the user is following, with a lot more visibility than the rest. This in turn benefits the content creator as they

need a certain number of followers and consistent viewers to monetize their channel.

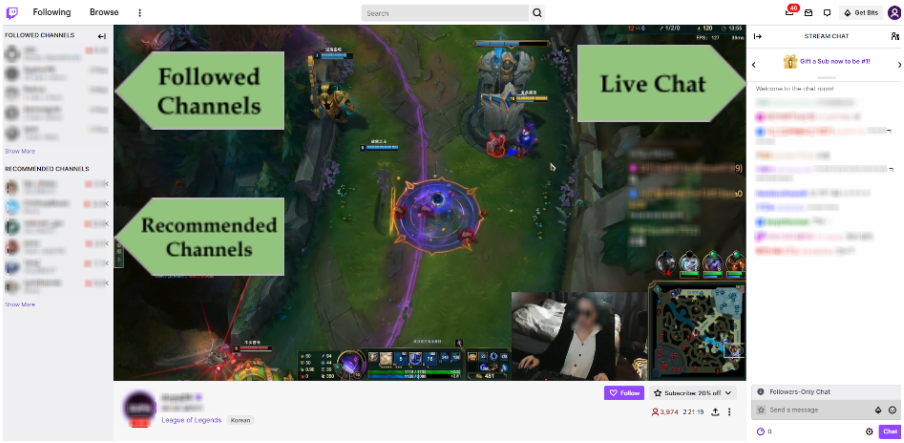


Figure 1: An overview of Twitch.tv platform

Given that live streaming channels are not explicitly rated, such as movies with ratings from 1 to 5 stars, users' viewings habits and followings can be used as implicit data to represent users' preferences. However, users' viewing time of channels can sometimes be misleading, for example, when a viewer engages in a channel for a long time for specific events. Users' followings might be a better measure of implicit ranking because it is a voluntary action that implies consideration and consistency from the user.

Users' followings data can be obtained via the twitch API, but this tool does not provide further information that can be used to enhance the interactions. Regardless, channels' metadata can be obtained via other publicly available sources, such as twitchtracker.com, an analytical website of live streaming channels.

1.3 *Motivation*

In regards to RS in the field of live streaming channels, there have been previous research using hybrid models, still, to the best of our knowledge, none of these studies has analyzed the performance of hybrid RS that combines collaborative and content data as latent representations of their features. An analysis based on this approach would not only provide an objective and detailed answer about the fitness of the model specifically for live streaming channels, but could also further evidence its robustness and generalizability as a recommendation tool. Satisfactory results would add to the claims that this approach can outperform specialized models (CF

and CB on their own) on real-world datasets with varying sparsity (Kula, 2015), and to its ability to be applied in uncharted domains.

Additionally, previous studies have mainly focused on users viewing time to score metrics and consider users' preferences towards live streaming channels, disregarding other relevant data, such as user's followings information. Using this type of interaction as the main input of users' preferences would be a novel approach, and for this reason, assessing its potential to predict users' preferences towards live stream channels is important and could lead to new lines of research.

For these reasons, the goal of this project is to develop and examine the performance of a hybrid latent representation recommender system for live streaming channels based on users' followings interaction data and other publicly available information.

1.4 *Research Outline*

The main research question that this project will address is described as the following:

> *To what extent can we implement an accurate RS of twitch channels using a hybrid latent representation recommender system based on users' followings and live stream channel's metadata?*

In reference to the main question, the following three sub-questions are derived to further describe the research outline:

RQ1 *To what extent does combining item features with interactions data improves the performance of a model based on latent representations?*

RQ2 *How does tuning hyperparameters, such as number of components, learning rate, and number of epochs affect the performance of the model?*

RQ2 *Does a hybrid RS based on latent representations outperforms other RS?*

To answer the questions above, the thesis will be structured as follows: Section 2 discusses the concepts and literature related to the topics of this project. Section 3 covers the methodological framework of this study, including a description of the model proposed, the datasets used, the evaluation methods, and the hyperparameters to be adjusted. The results of the analysis are shown in Section 4. Section 5 provides a discussion of the results in the context of the existing research. Lastly, in Section 6 the conclusion of the project in regards to the research objectives is formulated.

2 RELATED WORK

This section describes the previous research related to the main elements of this paper. Starting with a view of RS in general, passing through both categories of RS: the collaborative filtering and the content-based, and finishing with the previous research on hybrid RS, RS based on latent representations and RS specifically for live streaming channels.

2.1 *Recommender Systems*

Recommender systems can be described as tools that can use the data of the items they intend to recommend and the target users to create notions of affinity between them and generate meaningful and useful recommendations. Nowadays, RS are applied across multiple and very varied industries, recommending diverse items such as books (Rana & Jain, 2012), news (Raza & Ding, 2019), music (Gunawan, Suhartono, et al., 2019) and movies (Walek & Fojtik, 2020), just to name a few; and their specific designs are closely related to the domain and type of data for which recommendations are to be made (Melville & Sindhwani, 2010).

According to Sharma and Singh (2016), although there was already an independent line of research on recommendation systems as early as the 1970s, the first commercial recommendation system in operation was registered in 1990. This was intended to be able to filter out unnecessary emails and a collaborative filtering system was used to achieve this.

These statistical and mathematical methods previously relegated to technical experts became more popular thanks to the Netflix Prize. It consisted of a competition hosted by Netflix where the first team that could obtain a score 10% better than the system that the company was using at that time, would win one million dollars. The winning team used a linear combination of multiple approaches, which in turn, were hybrids between other classical models improved by customized techniques (Bell & Koren, 2007).

According to Melville and Sindhwani (2010), most of the RS approaches can be considered to be into one of three categories previously mentioned: collaborative filtering (CF), content-based recommenders (CB), and hybrid approaches. Further details of these categories will be given in the next subsections.

2.2 *Collaborative Filtering*

For CF approaches, recommendations are based on the decisions that other users who have similar preferences have made. The reasoning behind

this method is that people who agree or have the same taste about a set of items will probably have a similar agreement about other items too (Schafer, Frankowski, Herlocker, & Sen, 2007). To illustrate, if a user has rated 5 stars the movie "GhostBusters", the model would use the rating information of other users who have rated this movie similarly, and check which other movies have these users rated highly, and the target user has not watched yet, to make the recommendations.

This approach can take values in a scale, such as the 1 to 5 stars commonly used to rate movies (Ouyang, Liu, Rong, & Xiong, 2014), or binary such as a positive or negative interaction with a user in a social network (Cai et al., 2010). Usually, this type of binary input is associated with implicit data, such as videos watched, books lent, items bought, etc. (Verstrepen, Bhaduriy, Cule, & Goethals, 2017)

CF recommender systems are probably the most popular and widely implemented designs in the industry because their main strength is to be relatively agnostic to the domain in which they perform. This means that they can work well regardless of the complexity of the representation of the objects to be recommended, as long as the variation in user's taste has some correlation (Burke, 2002).

Nevertheless, CF has also several disadvantages. Because this type of recommender depends on the user's past decisions, its performance is especially faulty for users who have rated a low number of items or none at all, and for items that have not been rated by many users. This is commonly known as the Cold Start problem, for users and items respectively (Lam, Vu, Le, & Duong, 2008). Additionally, CF does not work well when data is very sparse, i.e. when few users have rated the same items, and for the so-called "grey sheep" users, referring to those users who have a particular set of preferences distant from any other group of users, so the recommendations presents no benefit for them (Burke, 2002).

2.3 *Content Based Recommender Systems*

The main inputs of this type of RS process are attributes or properties that represent in some way the set of items (Lops, Gemmis, & Semeraro, 2011). With this information, the model makes recommendations based on the similarity of the items the target user has liked in the past, by trying to match the previous selection of items of this user to new unseen items on these feature values (Aggarwal, 2016). To illustrate, if a user previously liked Interstellar, a Science Fiction genre movie directed by Christopher Nolan, the model might recommend other Science Fiction movies or movies directed by this same director, if no other movie meets both criteria.

Because CB recommender systems are independent of other users' ratings, they can recognize niche tastes and make recommendations that not a lot of people would prefer, and they also work well for new items or items that have not been very popular. But because they need to extract machine-readable features from the items to be recommended, in some domains, the content to analyze is either very scarce or hard to extract in a way that represents the object. This limitation can lead to overspecialization, where the model recommends the same items without displaying much novelty (Khusro et al., 2016).

2.4 *Hybrid Recommender Systems*

These approaches combine the previous methods into one, intending to bolster their respective advantages and reduce their disadvantages so the final output can outperform the base methods when applied on their own (Çano & Morisio, 2017).

Most of the combinations between the CF and CB systems are done by aggregating the output of the models, for example, by weighting (Miranda et al., 1999) the results of different models to produce a unique recommendation. Other techniques can be mixing results from different recommender systems and presenting them altogether (Smyth & Cotter, 2000), or "cascading" the results of one RS to refine the recommendations of the final one (Chen, Niu, Zhao, & Li, 2014).

Another effective way of hybridization is by combining features instead of the outputs, to feed one single model that can unify them, and make recommendations based on the combination of these data sources. In recent years, there has been an upsurge of these designs, specially in the field of matrix factorization, where the interactions typically used for CF models are enriched with side information to help tackle the hurdle of data sparsity and cold-starts (Barjasteh, Forsati, Ross, Esfahanian, & Radha, 2016).

2.5 *Hybrid Latent Representation Models*

Although there are other methods based on latent representations, such as, deep learning ones (Zhang, Yao, Sun, & Tay, 2019), this paper focuses on Matrix Factorization (MF) as it is the basis for the proposed model. As a general principle, MF models learn low-rank representations (also called latent representations or latent factors) of the user-item interaction matrix that is then used to predict scores for unknown item-user interactions. Because this approach presents high scalability power and accuracy, and it

can also integrate side information with the collaborative data, it has been extensively studied (Shi, Larson, & Hanjalic, 2014).

The first attempt of MF enhanced by side information applied to the field of RS can be traced back to the work of Singh and Gordon (2008), where the matrix of interactions is collectively factorized with the side information matrix. Several other variations have been applied to different fields, such as movies (Shi, Larson, & Hanjalic, 2010) recommendations or landmarks to visit (Shi, Serdyukov, Hanjalic, & Larson, 2011) and product reviews (Ma, Lyu, & King, 2009). However, to the best of our knowledge, this approach has never been applied to live streaming channels recommendations.

2.6 *Recommender Systems for Live Stream Channels*

Live streaming in general, and twitch.tv in particular, have had a great influence on the entire video game industry (Johnson & Woodcock, 2019). Because of this, numerous studies have been conducted based on this platform, but most of them fall on disciplines other than those of machine learning and artificial intelligence. For this reason, it comes as no surprise that research literature in the area of recommendation systems specifically for live stream channels is not extensive, but there have been some previous relevant work that will be described next.

Yang, Shih, Huang, Ting, and Liu (2013) have created a hybrid recommendation system that combines the results of a CF model with a CB model by weighting the results of the CF based on similarity computed by the CB model. So the channels with more similar features to the the ones watched by the target user, and the channels watched by users with similar features to the target user, will have a bonus rating score of the results from the CF. Additionally, they combine the final result of this hybrid model with another separate algorithm, called Most Recent View (MRV), which is responsible for searching for the most recent viewing pattern of the target user and recommending the most frequent channels. These two different models switch with each other based on the type of viewer they have to serve: if the viewer does not have a high amount of channels watched in their record, or usually frequents the same ones, the RS will use the MRV algorithm, otherwise, the first method described will be used.

Liu, Lin, and Huang (2015) used an algorithm called HITS that was developed by other researchers to extract information from the network structure of a hyperlinked environment (Kleinberg, 1999). With this algorithm, they quantify the "authority" values of the channels and sort them via this criteria, arguing that the higher this score is, the better the channels are. Finally, the highest ranked channels are recommended to clusters of

users, that are grouped using the k-means clustering technique based on the preferences of each user on each channel.

Lin, Chen, Chen, and Chen (2021) made a more personalized recommendation system using an algorithm called n-Most Similar Neighbors. Concisely summarizing this method, it consists in searching for the n users with the most similar tastes to that of the target user and recommending the preferences of these n corresponding users. However, instead of using other classic similarity measures, they introduce the Rank Ordering Clustering algorithm to order users with similar behavior more efficiently. According to the results of their experiments, this model is the one that achieves the best result compared to previous models.

3 METHODS

The following chapter describes the methodological aspects of the paper in order. The process of obtaining the raw data is explained and the processing steps to transform this data into the final versions of the datasets to use in this paper are reported first. Subsequently, a description of the proposed hybrid RS is presented, followed by the framework to evaluate the models. Next to that, the hyperparameter to be optimized are presented, and to conclude, a diagram of the workflow is presented, and a brief description of the software tools used are also included.

3.1 *Obtaining the Raw Data*

3.1.1 *Interactions Dataset*

Twitch.tv public API was used to obtain the interactions dataset between viewers and channels on users' followings. To obtain this data, three different endpoints from the API were used:

1. **Streams:** *provides data of the top n channels, ordered by the number of active viewers, at the time the endpoint is called*
2. **Follows/to_id:** *returns n random users following a stream channel given its specific ID*
3. **Follows/from_id:** *returns n random stream channels that a user is following given its specific ID*

The process started by calling the ***Streams*** endpoint to retrieve the top 100 channels every 4 hours per day during the entire second week of October of 2021 until 1000 different of the most popular live stream channels were accumulated. Subsequently, for each of these channels, the endpoint

Follow/to_id was called to obtain 100 users following each channel, with features such as username, and DateTime they started following the respective channel. Finally, ***Follow/from_id*** was called for each of the different users obtained by the last endpoint, to retrieve all the channels that these users were following. This resulted in more than 4 million interactions between viewers and live streaming channels.

3.1.2 *Channels' Metadata*

Channels' metadata was obtained via scraping www.twitchtracker.com, an analytic website with data about popular live streaming channels, during the month of November 2021. The search of channels was done using the names previously obtained from the Twitch.tv public API. Channels with characters from different alphabets were not included because of difficulties to be matched. From this process, we obtained almost 10.000 unique live stream channels with numerical, such as the number of followers and total hours the channel has been live; and categorical features such as language, most played games, and date of creation, to list a few.

3.2 *Pre-Processing and Final Datasets*

3.2.1 *Interactions Dataset*

The first step of cleaning the interactions dataset was to remove duplicates created from the process used to query data from the API because the data for each user obtained from calling the first endpoint would be repeated when calling the second endpoint. This greatly reduced the length of the dataset.

Afterwards, all the interactions to live streaming channels that were not present in the channels' metadata set were removed. This was done for the hybrid RS to work properly, as it needs features for each channel that is present in the interactions.

Lastly, we subset the data by removing users with few interactions using a minimum threshold of 15 interactions, i.e., users following less than 15 channels were not included. This value was set arbitrarily after some exploration of the dataset.

The result was a dataframe of 1.3 million interactions, with 33835 unique users, following an average of 40 channels each, with the minimum being 15 followed channels and the maximum 121; and 9788 unique live stream channels, having an average of 138 followers each, with the minimum being 5 followers and the maximum 6391. A graphic depicting the cumulative distribution for the top 1% mos followed channels can be observed in Appendix A. The columns present are:

- **username_id:** *ID that twitch.tv has given the user that is following a live stream channel*
- **username:** *username of the user that is following a live stream channel has in twitch.tv*
- **channel_id:** *ID of the channel that twitch.tv has given to the channel that the user is following*
- **channel:** *channel name in twitch.tv that the user is following to*
- **followed_at:** *datetime indicating when the user has started following the channel*

3.2.2 *Channels Metadata*

The processing of the channels' metadata started by removing the columns related to subscriptions data, given that a large number of channels were missing this information

Categorical features were transformed using one-hot encoding so they could be used by the models. This method consists in creating a different column for each value of the categorical variable and assigning these columns a 0 or a 1 depending on if the value of the category is present. Numerical features were scaled down to 0, 1 and 2 depending on if the values fell inside the lower tail, middle or higher tail of the distribution of each variable respectively. This was done because the models to be used are affected by the range of the features.

The result was a dataset with 9788 different channels and 410 features. The distribution of the categorical features can be observed in Appendix B Following, there is a brief description of the features utilized:

- **channel name:** *name of the channel in twitch.tv*
- **total hours:** *total amount of hours the live stream channel has been live*
- **hours watched:** *total amount of hours that the channel has been watched (we assume it is calculated by summing up the time each user has spent on the channel)*
- **total views:** *total amount of users that have seen the channel (not unique users)*
- **total followers:** *total amount of followers the channel has*
- **record viewers:** *highest record of concurrent viewers in a live stream session*

- **record viewers:** *highest record of concurrent viewers in a live stream session*
- **language (One-Hot Encoded):** *main language of the channel*
- **year created (One-Hot Encoded):** *year the channel was created in twitch.tv*
- **record viewers:** *highest record of concurrent viewers in a live stream session*
- **average viewers:** *average amount of concurrent viewers in the channel's live stream sessions*
- **active days:** *total amount of days the channel has been live*
- **average hours:** *average amount of hours each live stream session lasts*
- **active days per week:** *average of days per week the live stream channel is live*
- **average games:** *average number of games the streamer plays in a live stream session*
- **most played games (One-Hot Encoded):** *name of the game the streamer has played the most*

3.3 *Algorithms*

In this subsection, a brief introduction of Matrix Factorization methods, as the basis of the proposed model, is presented. Afterwards, the algorithm to be analyzed is described.

3.3.1 *Matrix Factorization for Collaborative Filtering*

Some of the most popular CF models, and the one used in this paper as the basis of the hybrid RS, are based on matrix factorization (MF).

The basic concept of matrix factorization is to break down a matrix into smaller and simpler pieces without losing its original values. It is commonly used as a dimensional reduction and feature learning method (Babaee, Tsoukalas, Babaee, Rigoll, & Datcu, 2016) where a matrix gets decomposed into its constituent parts so that the original values are maintained but operations can be easier to calculate.

Nevertheless, when this method is used as a RS, we face the issue of missing elements inside the matrix. These missing elements represent the users and items that have not interacted yet, and for this reason, these are

the values we are most interested in. The goal of this technique is then, to use the elements that are present in the interactions matrix to "learn" the best value for the missing ones, such that the inner product of mapping of items and users best approximates the values in the original matrix.

The most basic MF for RS leverages latent factors, that is, inferred patterns from the data. In this case, matrix R of n users $\times$ m items is decomposed by K latent factors and we end up with two matrices $P_{n\times k}$ and $Q_{k\times m}$. These discovered factors might represent obvious dimensions of the users and items, such as a category of games for channels or an age group for users, and the decomposed matrices represent the strength of the relationship that users and items have within these latent factors (Koren, Bell, & Volinsky, 2009).

These latent patterns are learned by initializing both matrices P and Q with random values and improving the approximation by minimizing or maximizing an objective loss function based on the existing values using a specific learning algorithm.

As an illustration, let R be a dataset composed of item ratings given by a set of users, where the target rating to predict is for User 1 (Row 1) and item 3 (Column 3), represented as "?".

$$\begin{array}{c} \\ \text{Users} \end{array} \begin{array}{c} \begin{array}{ccccc} & & \text{Items} & & \end{array} \\ \begin{bmatrix} 4 & 2 & ? & 2 & 3 \\ 5 & 1 & 2 & 3 & 5 \\ 3 & 1 & 3 & 2 & 1 \\ 4 & 4 & 3 & 1 & 2 \\ 2 & 1 & 5 & 4 & 2 \end{bmatrix} \end{array} \tag{1}$$

With Root Mean Squared Error (RMSE) as the loss function, gradient descent as the learning algorithm and K (latent factors) = 2, the matrix R would be decomposed as the following:

$$\begin{array}{c} \text{Users} \\ \\ \end{array} \begin{array}{c} \begin{array}{cc} & \text{K} \end{array} \\ \begin{bmatrix} 1.3441 & 1.1330 \\ 0.5428 & 2.2134 \\ 2.0622 & 0.2371 \\ 1.0652 & 1.284 \end{bmatrix}_P \end{array} \times \begin{array}{c} \text{K} \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & & \text{Items} & & \end{array} \\ \begin{bmatrix} 1.5674 & 1.6918 & 1.7583 & 0.5253 & 0.4873 \\ 1.5231 & -0.1857 & 0.9853 & 1.5324 & 1.8477 \end{bmatrix}_Q \end{array}$$

Then, we can obtain the original values of matrix R (or their approximations) as a dot product of matrices P and Q, that includes the previous missing element and its estimation through this method:

$$\text{Result}$$
$$\begin{bmatrix} 3.8325 & 2.0636 & 3.4797 & 2.4424 & 2.7485 \\ 4.2223 & 0.5073 & 3.1355 & 3.6772 & 4.3544 \\ 2.7098 & 1.6153 & 2.5221 & 1.6396 & -1.8260 \\ 3.5935 & 3.4448 & 3.8597 & 1.4467 & 1.4431 \\ 3.6322 & 1.5629 & 3.1426 & 2.5342 & 2.8999 \end{bmatrix} \tag{2}$$

3.3.2 *Hybrid Latent Representation Recommender System*

The hybrid latent representation algorithm proposed by Kula (2015) uses the same principles of MF for collaborative filtering but instead of applying it to the matrix of Users and Items directly, it does so to the linear combinations of their constitutive features. So, for example, a movie item "Interstellar" that can be described by features such as "Genre": "Sci-Fi", "Director": "Christopher Nolan" and "Main Role": "Matthew David McConaughey" will be represented as the sum of these features, or a user with ID = 50440 can be represented as the sum of "Gender": "Female", "Location": "China", "Age": "22".

The advantage of using the representations of users and items based on their constitutive features is that, most of the times, the features will be fewer than the user and items themselves, so this results in a much more dense matrix with fewer parameters to be estimated, which helps reducing risks of overfitting and making it more capable of generalization. In addition, classic CF models have problems performing for cold start items and users when represented only as indices because such elements need enough accumulated past interactions data to be estimated accurately, whereas, in this model, users and items can be estimated reasonably good from the start by representing them as a combination of their features and estimating the features' latent representations.

The model is formally described by Kula (2015) as follows. For a set of Items I and a set of Users U that can interact either in a favorably (S^+) or a negatively (S^-) way, let the set of all item-user interactions $(u, i) \in U \cdot I$ be the union of these possible interactions. Users and items are described by their features, so each item i can be described by the set of features $f_i \subset F^I$ where F^I is the total set of features for I, and the same holds for each user.

The model is parameterised in k dimensional item and user feature latent representations k_f^I and k_f^U for each feature f. The features are also described by a scalar bias term b_f^I for item features and b_f^U.

After each feature is corresponded to a parameter vector and a scalar bias term, the latent representations of a given item i is can be computed as the sum of its latent vectors:

$$p_i = \sum_{j \in f_i} k_j^I$$

The same holds for user u:

$$q_u = \sum_{j \in f_u} k_j^U$$

And the bias term for item i is given by the sum of the biases associated to its features:

$$b_i = \sum_{j \in f_i} b_j^I$$

And the same holds for the user u:

$$b_u = \sum_{j \in f_u} b_j^U$$

The probability of interactions $\hat{r}$ between a user u and item i is given by the dot product of the user and item representations vectors, adjusted by users' and items' feature biases

$$\hat{r}_{ui} = f(p_i \cdot q_u + b_i + b_u)$$

And in this case, transformed via a sigmoid function, because the prediction is intended to be binary (follow or no follow a channel)

$$f(x) = \frac{1}{1 + \exp(-x)}$$

The specific method for optimization used in this model is called Learning-to-Rank and was first introduced by Weston, Bengio, and Usunier (2011) and later developed by Weston, Yee, and Weiss (2013) for RS, and it is particularly useful when the goal is to optimize the Precision@k

scores. Following next, a brief and concise explanation of this algorithm is presented.

The algorithm samples random positive user-item labeled pair, that is, an interaction the model knows has happened, and a random negative item for the same user. Then computes the prediction for both items given the values of the lower dimensionality matrices, and performs a correction (gradient update) on the parameter of the model if the true ranks are violated, meaning, if the known negative item prediction exceeds by a margin (in this case, margin = 1) the prediction for the known positive item. If the ranks are not violated, then the algorithm samples another random negative item iteratively either until the ranks are violated or a given cutoff is reached. The magnitude of the update is based on the effort the algorithm made to find a violating rank sample. If the prediction of the model was wrong in the first attempts, it could be an indication that many negative items are being ranked higher than positive items in the actual state of the model, thus the update needs to be larger. Conversely, if the model made numerous iterations to find a rank violation, this is probably because the values are optimal, and consequently, the updates need to be lower.

3.4 *Baseline Models*

Each of the three baseline models used to compare the proposed model is described in this subsection. Starting with a version of a pure CF based on latent representation is presented as a simpler variation of the proposed model. Following, a simple version of a CB recommender system is described, and lastly, the algorithm based on Most Popular Channels is presented.

3.4.1 *Pure Collaborative Filtering*

The model proposed by Kula (2015) based on latent representations (see Section 3.3.2) can only be considered hybrid if, besides data of the interactions between users and items, features of either entity are also presented. Otherwise, the model is reduced to a classic Matrix Factorization model where items and users are represented as their indicator feature in the interaction data (e.g., user ID).

$$\hat{r}_{ui} = f(I_i \cdot U_u + b_i + b_u)$$

For this reason, a model provided only by indicators (an ID number) for users and items will be the collaborative filtering baseline to compare the hybrid model.

3.4.2 *Pure Content Based Recommender System*

For this paper, a simple CB recommender system was developed as another baseline model to be used as a comparison. The algorithm consists of taking the last followed channel for each user and making a nearest neighbor search based on the similarity of the features, i.e., searching for the channel with the most proximate feature values. To measure proximity, the Euclidean distance is used.

This type of distance measures the length of a line between two points. So the top six recommendations for a user are given by the six closest nearest neighbors based on the distance of the features using the following formula:

$$D(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3}$$

This algorithm assumes that users' preferences get refined with time, and the more time they spend on the platform the more they acquire a specific taste. This is the reason the last followed channel of each user is chosen as the anchor point, as it might have more similarities to other channels in their list of followed channels, in comparison to a randomly selected channel of this list.

3.4.3 *Most Popular Channels (MPC) Baseline*

The baseline model to compare the absolute performance of the hybrid latent representation RS is a popularity-based model. This is a common method to set a benchmark performance (Ji, Sun, Zhang, & Li, 2020) consisting of sorting the items based on the frequency of interactions and using the top-k most interacted with as recommendations for every user.

3.5 *Evaluation*

Here is presented the overall framework to fairly and consistently evaluate the performance of the proposed model and the baselines. First, the specific scoring metric is explained. Then, a description of the dataset splits that are implemented to validate the final scores is stated, and finally, a specific split made to evaluate the performance of RS in consistency with their goals is explained.

3.5.1 *Evaluation Score*

As is the case for entities dealing with a lot of data, twitch.tv can only show a limited amount of recommendations for each user before these start to be counterproductive. For this reason, the scoring function use to evaluate the RS in this study is Precision@k. This is a very common metric for recommender systems that measures the proportion of the top k recommended items that are relevant to the user (Kar, Narasimhan, & Jain, 2015). In this case, a relevant channel for a user would mean a channel that they are willing to follow.

$$\text{Precision@k} = \frac{\text{Relevant Items}}{\text{Observed Items}}$$

The Precision@K for the entire dataset is the sum of the scores for each User divided by the number of users. The value of K to use for this paper is 6 and 10, because twitch.tv shows 6 recommendations as a default, and they present the possibility to expand the recommendations up to 10.

3.5.2 *Train & Test Split*

To test the scores of the final configuration of the model, the interactions dataset was partitioned into a training and testing set, with the latter being a subset of 20% of the total **users** with their respective interactions data points. The dataset for item features was not partitioned in any way, given that all channels' features need to be present to evaluate the hybrid RS.

3.5.3 *Hidden Interactions Split*

Because the goal of a RS is to recommend unseen items for the users, an additional step is needed to evaluate the model accurately. The set to be evaluated is partitioned into two disjoint sets, but in this case, the partition is made on the basis of users **interactions**. This is useful because we can remove the items used to train the model from the set of potential recommendations and score the model's recommendations based only on the items present in the set with "hidden" interactions. By doing this, the scoring of the models is better aligned with the goal of RS and the real behavior of users (Shani & Gunawardana, 2011). The percentage of interactions hidden for both train and test sets is 40%, given that the minimum interaction threshold applied was 15 interactions, this percentage represented a minimum of 6 interactions for each user in the hidden set allowing a more fair Precision@6 scoring for all users.

Following, a diagram with the two different splits performed on the interactions dataset is shown for more clarification.

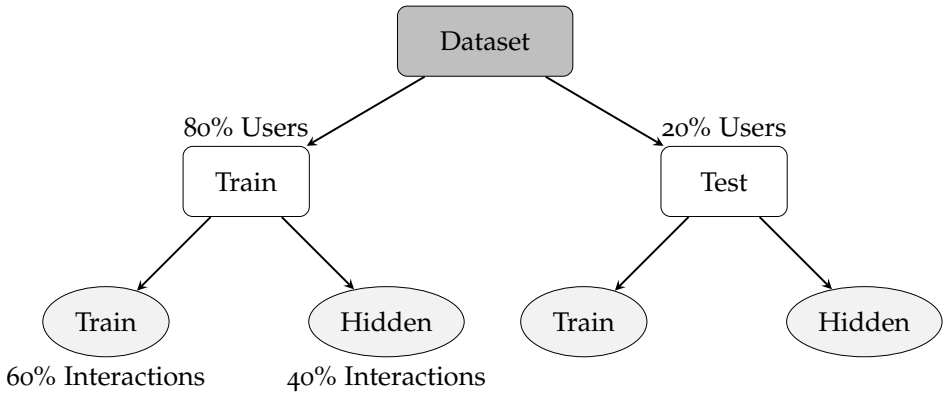


Figure 2: Diagram of datasets splits

3.6 *Hyperparameters*

Machine Learning algorithms have a number of parameters that, in contrast to first-level parameters such as weights and bias terms, they have to be configured before the training of the model starts, and they remain constant throughout the process. These second-level parameters are commonly known as hyperparameters, and they can be optimized to achieve a better performance of the model (Probst, Boulesteix, & Bischl, 2019). However, depending on the size of the dataset at hand, the nature of the algorithm, and also the computational resources, the search for the optimal configuration can take a considerable amount of time (Claesen & De Moor, 2015).

Taking into consideration the scope of this paper and the limitations of resources and time, only three of the most relevant hyperparameters for this model are tuned: number of components, learning rate, and number of epochs. The method to use for an optimal search of hyperparameters is random search, because it has been proven empirically and theoretically that it can be a very efficient method (Bergstra & Bengio, 2012). Due also to limitations of time, only for this task, a reduced dataset is used that consists of a random selection of 20% of users from the training set. The hyperparameters are tuned based on the scoring metric Precision@6, this is to say, the configuration of the model that reaches the best Precision@6 score in the training set will be used to give the final score in the test set. The three hyperparameters chosen to be tuned are described in more detail next:

3.6.1 *Number of Components*

This represents the dimensionality of the feature latent representations. The right number of components depends on the underlying properties of the phenomenon being investigated, and it is often not known. If k is set too small, the potential patterns in the structured data are missed and the goal of dimensionality reduction is not achieved. Whereas if k is too large, these relevant components may become very fragmented and very difficult to interpret (Maisog et al., 2021).

3.6.2 *Learning Rate*

The learning rate controls the size of the change to the model after the error estimation. This is a very important parameter given that it relates directly to how quickly and well the model converges to the optimal solution. The smaller the learning rate is, the more time it takes the model to reach the global minima and can even become stagnant. On the other hand, if the learning rate is too big, it will converge faster but to potential sub-optimal solutions.

3.6.3 *Number of Epochs*

The number of epochs can be set from one to infinity, and it determines the total amount of times each sample in the training set has had the opportunity to be updated. Typically, the number of epochs is set to a high number, because it helps to reduce training loss at the cost of more training time. Nevertheless, setting this parameter too high can lead to the model overfitting on the training dataset, but setting it too low might cause low accuracy scores due to the model not having enough training attempts to adapt.

Fifteen different configurations of these parameters are combined randomly and compared. For each model configuration, random values between ten and a hundred are drawn to set the number of components (K), values between five and fifty are also randomly assigned for the number of epochs, and as for the learning rate (η) a random sample from an exponential distribution parameterized as:

$$f(x; \frac{1}{\beta}) = \begin{cases} \frac{1}{\beta} \exp(-\frac{x}{\beta}) & x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

With the value of the scale parameter β set to 0.05 is assigned. All other hyperparameters of the model based on latent representations are set to the default values from its implementation package.

3.7 *Work-Flow*

For clarification, a work-flow illustration of the processes for this paper is presented following.

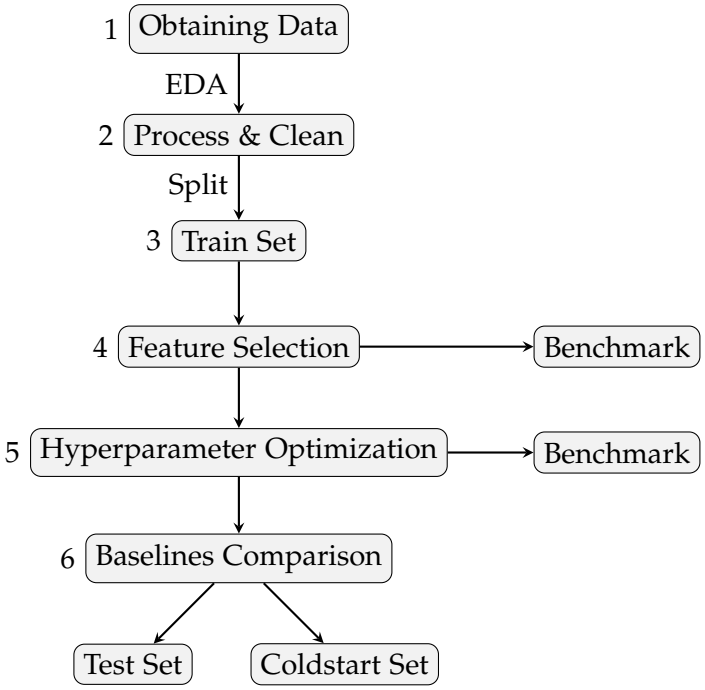


Figure 3: Work-Flow Diagram

The first two parts are explained in Section 3.1, Section 3.2 and Section 3.5.2. Once we have the data partitioned in train and test sets, a feature selection process to evaluate the impact of adding side information is carried out with the train set. Afterwards, hyperparameter optimization is carried out with the selected features to obtain the best performing model. Finally, the proposed model is compared against the baseline models with the test set and a "cold-start" set (see Section 4.3). The entire process implemented, and the respective data, can be revised at the following repository: https://github.com/lautaropacella/DS-S-Master-Thesis

3.8 *Software*

The data collection, analysis, and comparison of the models are carried out using Python (3.9.7) programming language. Different packages are used to perform different technical tasks. The data collection is done using the Request and BeautifulSoup packages. The library Pandas is used to analyze and manipulate the raw data. LightFM package is used to create the baseline collaborative filtering model and the hybrid latent representation model. Lastly, SciPy provides the functionalities to create the baseline CB recommender system. Data collection is performed using Virtual Studio Code as the integrated development environment, and the remaining tasks are done in Jupyter Notebook.

4 RESULTS

The first part of this chapter describes the feature selection process carried out and the analysis of the version of the model with added features against the pure collaborative filtering version where no item feature was added. The second part is related to the hyperparameter search and its influence on the hybrid latent representation RS. The last section exhibits a comparative analysis of the best version of the proposed model against all the baseline models.

4.1 *Side Information Addition*

In order to answer the first specific research question proposed in this paper related to adding side information to a model based on latent representation and to what extent this can improve its performance, firstly, a naive approach was tested, using a pure collaborative filtering version of the model with no item features included as the benchmark to compare a version of the hybrid version of the model with all side features added. The results of their performances at Precision@6 and Precision@10 are shown next.

Table 1: Comparison of Precision@6 and Precision@10 scores for baseline model and model with all item features added.

	Scores	
	Precision@6	Precision@10
Baseline	**0.11**	**0.10**
With All Item Features	$4e-5$	$3e-5$

It is evident that including all features without discrimination severely undermines the performance of the baseline model that only includes interactions data, resulting in Precision@6 and Precision@10 scores nearing 0. For this reason, a feature selection process was required to obtain the best performing version of the hybridized model.

Using the same previously mentioned model as a baseline, a simple process of feature selection was carried out by comparing this baseline to versions of the same model but with the addition of each feature available one by one (separately, not cumulative): Total Hours (TH), Total Views (TV), Total Followers (TF), Record Viewers (RV), Hours Streamed (HS), Hours Watched (HW), Average Viewers (AV), Active Days (AD), Average Hours (AH), Active Days Per Week (ADPW), Year Created (YC), Language (L) and Most Played Games (MPG). This comparison is presented in the following graph.

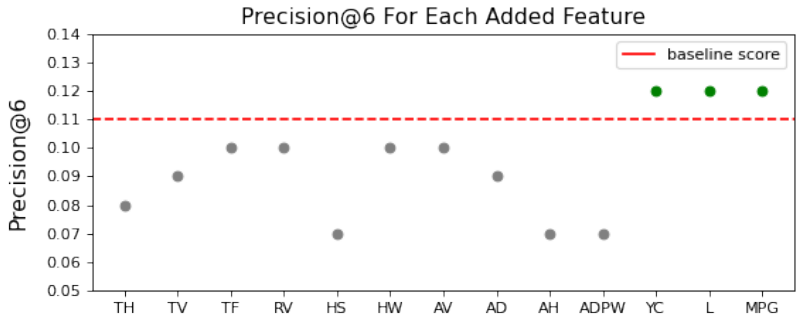


Figure 4: Comparison of Precision@6 scores for each feature added to the baseline model

From these results, it can be observed that only the three categorical features yielded better scores when they were added to the baseline model, being Year Created (YC), Language (L), and Most Played Games (MPG) all three with a Precision@6 Score of 0.12.

As a heuristic for further inspection, only these three features that resulted in better scores than the baseline model without including them were used to compare different versions of the models containing all possible combinations of these features. The results of these comparisons are presented in the next table.

From the comparison of the different versions of the model with the combinations of the categorical features (YC, L & MPG), it is possible to observe the best possible combination is obtained by only including Language & Most Played Games as item features for the model, presenting the highest scores of Precision@6 (0.15) and Precision@10 (0.13) alike. The worst performing combination of these features includes Year Created &

Table 2: Comparison of Precision@6 and Precision@10 scores for combinations of baseline model with selected features.

		Scores	
		Precision@6	Precision@10
Baseline		**0.11**	**0.10**
Combinations	L & YC	0.12	0.10
	L & MPG	**0.15**	**0.13**
	YC & MPG	0.10	0.09
	L & YC & MPG	0.14	0.12

Most Played Games, with a Precision@6 score of 0.10 and a Precision@10 score of 0.09. It can also be noted that including all three features together into the model did not provide the best resulting score.

The best performing version of the model that includes only the features Language and Most Played Games was contrasted against the baseline model also with the test set. The final scores are depicted in the following table.

Table 3: Comparison of Precision@6 and Precision@10 scores of model with selected features against the Most Popular Channels (MPC) baseline algorithm.

		Scores	
Set	Model	Precision@6	Precision@10
Train	Baseline	0.11	0.10
	Hybrid	0.15	0.13
Test	Baseline	0.10	0.08
	Hybrid	0.14	0.12

The hybrid latent representation model using the selected features scored better than the baseline CF model in both settings (training and testing set) and both metrics (Precision@6 and Precision@10). In general, the differences in the performances are not very sizable. In the training set, the difference between the models for Precision@6 is only 0.03, and 0.02 for Precision@10 in favor of the hybrid model. For the testing set, the supremacy of the hybrid approach becomes more apparent, with a lead of 0.04 points both in Precision@6 and Precision@10.

4.2 *Hyperparameters*

To assess how the hyperparameter configuration can influence the performance of the proposed model, as described in section 3.6, 15 different combinations of the hyperparameters **number of components (k), learning rate (η) and number of epoch** were carried out using the random search optimization based on Precision@6 scores on a reduced dataset. The scores for each iteration are presented in figure 5, and the specific values for each hyperparameter can be seen in appendix C. The range of the scores varies between 0.229 and 0.271, indicating that hyperparameters can have an impact on performance. The maximum score was obtained in the 5th iteration, and the minimum score appears in the 13th iteration, but no analysis can be done based on this pattern, given that the method for hyperparameter search is based on randomness. The relation between each hyperparameter and the Precision@6 score is discussed next.

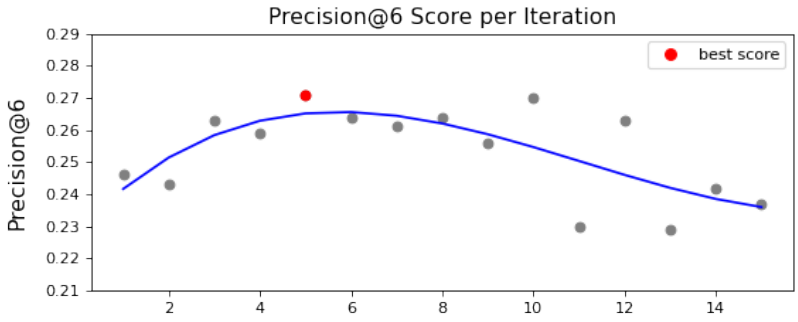


Figure 5: Comparison of Precision@6 for each Random Search Iteration

4.2.1 *Number of Components*

Figure 6 shows the results for configurations of the Number of Components (K) in 15 iterations. It can be observed that the best result was obtained with a K value of 90, and the lowers Precision@6 was the result of using 50 components. Nevertheless, no clear-cut trend can be extracted from the graphic.

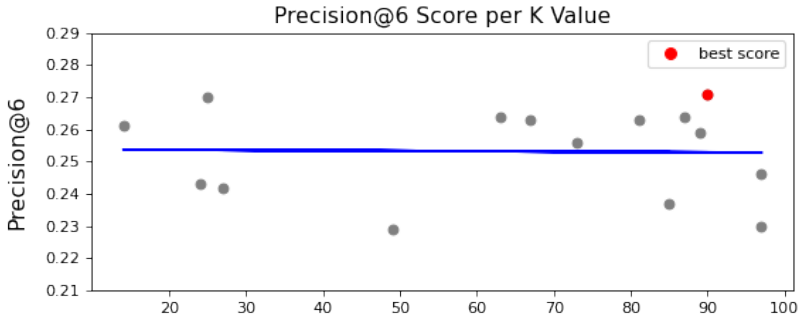


Figure 6: Comparison of Precision@6 for each K Value

4.2.2 *Learning Rate*

The results of tuning the Learning Rate (η) are displayed in Figure 7. It shows that the best score is obtained when setting the Learning Rate close to 0.02 (0.0230 exactly), and the highest value for η (0.150) produced the lowest score for Precision@6.

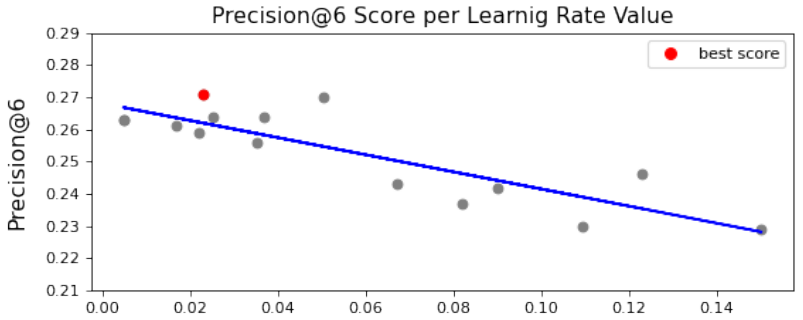


Figure 7: Comparison of Precision@6 for each η Value

This figure also shows a more evident negative correlation between the two variables. A Pearson R value of -0.85 ($P < 0.001$) suggests that the lower η is, a higher score for Precision@6 is expected for the proposed model.

4.2.3 *Number of Epochs*

As for the Number of Epochs, Figure 8 presents the results of the 15 different configurations for this value. It can be seen that the Precision@6 score was the result of 24 epochs, and the lowest score appeared when using 45 epochs.

A negative correlation can also be detected in this graphic, yet not so meaningful as the one for Learning Rate. A Pearson R value of -0.43 (P =

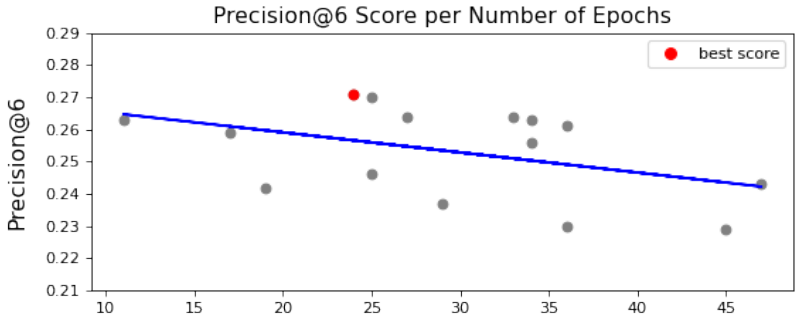


Figure 8: Comparison of Precision@6 for each Number of Epochs

0.10) for the correlation between the Number of Epochs and Precision@6 scores confirms this variable's lesser influence.

4.3 *Proposed Model Vs. Baselines*

Three versions of a model based on latent representations are presented in this paper. The most simple one, based only on interactions data as a pure CF recommender system is used as one the baseline for the other two versions of hybrid models based also on latent representations. One of the hybrid versions includes the features after the selection process (Hybrid + F) but does not use the best hyperparameter configurations found after the random search process. The third one (Hybrid +F+H) is the proposed model including the best scoring features and the best scoring hyperparameter configuration. These models are also compared to a pure CB recommender system (see Section 3.4.2) and an algorithm recommending channels based on popularity (MPC, see Section 3.4.3). These approaches were compared on the training and testing dataset, and one other setting. A third "cold-start" set, where the threshold for users with a minimum threshold of interactions was not applied (see Section 3.2. The results of comparing all the previously mentioned approaches in these three different settings are shown in the table below.

From the results, it can be observed that the proposed model with the selected features and best hyperparameter configuration performed the best in every setting and for both Precision@6 and Precision@10 scores. The worst performing model in all cases was the CB recommender system. The cold-start setting is shown to be the most detrimental to the performance of every model, still, the proposed model keeps achieving a Precision@6 score of 0.16 which, as a rough estimation, is one relevant channel out of six recommended per user.

Table 4: Comparison of Precision@6 and Precision@10 scores of model with selected features against the Most Popular Channels (MPC) baseline algorithm.

		Scores	
Set	Model	Precision@6	Precision@10
Train	Pure Collaborative Filtering	0.11	0.10
	Pure Content Based	0.01	0.01
	MPC	0.06	0.05
	Hybrid + F	0.15	0.13
	Hybrid + F + H	0.22	0.19
Test	Pure Collaborative Filtering	0.10	0.08
	Pure Content Based	0.01	0.01
	MPC	0.06	0.05
	Hybrid + F	0.14	0.12
	Hybrid + F + H	0.21	0.18
cold-start	Pure Collaborative Filtering	0.08	0.07
	Pure Content Based	0.01	0.01
	MPC	0.03	0.03
	Hybrid + F	0.11	0.09
	Hybrid + F + H	0.16	0.13

5 DISCUSSION

The first section of this chapter serves as a general discussion of the findings of this study. The limitations of these results are presented in the following subsection, and concluding, the impact and potential lines of research are discussed.

5.1 *General Discussion*

In regards to the influence of the channels' metadata as side information added into the base latent representation model to enrich the collaborative data, it can be observed that the best scoring features for the model (Language and Most Played Game) are in line with the ones used in previous research (Yang et al., 2013) as the main features to represent channels. This can be explained by the fact that the social aspects of live streaming are one of the main reasons for users to engage (Sjöblom & Hamari, 2017), hence a shared language that allows users to understand the streamer and communicate with them and the community becomes very relevant. Nevertheless, there is a specific type of audience that has an interest in particular games rather than the streamer or the community (Gandolfi, 2016), so for this type of audience, the dominant criteria for channel selection would be the game

it shows. Finally, as previous research indicated (Barjasteh et al., 2016), the addition of side information in the form of item features did have a positive impact on the model performance, although this impact was not as strong as it was expected. The results suggest that hyperparameter optimization has a bigger influence than adding side content information.

Concerning the results of the hyperparameter tuning process of this study, the expectations of a considerable rise in the scoring function for the proposed model were met, in accordance with the literature in the field (Galuzzi, Giordani, Candelieri, Perego, & Archetti, 2020). Moreover, this study also detected a strong negative correlation of the Learning Rate (η) with the score of the model and the weak influence of the Number of Components (K), found by Galuzzi et al. (2020). Studies suggest that the Learning Rate heavily influences the converging of the parameter space via Stochastic Gradient Descent (Jastrzębski et al., 2017)), and this learning algorithm is a crucial part of the optimization method used in the proposed model.

The comparison between the proposed model and the baseline models confirms the benefits of the hybrid approaches in general (Burke, 2002). Additionally, it specifically adds to the claims about the superiority of a hybrid latent representation model against pure CF and pure CB recommender systems in a context of varying sparsity, as well as in a "cold-start" setting (Kula, 2015). In addition to this, it can be observed that the difference between the CB model and the CF is much greater than that of the CF with the proposed model. Although the superiority of the CF was expected (Thorat, Goudar, & Barve, 2015), such a low score for the CB recommender system was not, being surpassed even by the algorithm based on the popularity of the channels.

5.2 *Limitations*

Because this study uses an innovative source of collaborative data (users' followings instead of users' viewing time), it lacks a true competitor to make a fair comparison. The final scores obtained can be regarded as low on their own, but high when compared to the baseline models used in this study. Assessing the true implication of the final scores of the proposed model becomes a difficult task without having previous studies examining this source of data. Yet, it is important to note that the prediction is done for channels that a user is willing to **follow**, and this type of interaction is usually a long standing one, so the value of having 1 relevant channel for each 6 recommended ones per user can be much higher a higher predictive score with other more volatile users' preferences measures, such as viewing time.

Another relevant constraint for this study was the time and computing resources available. This influenced many of the steps of the study, such as the processing techniques used to transform the variables and the feature selection process, or the method chosen to optimize the hyperparameters as well as the number of hyperparameters to be optimized. Counting on more of both of these resources would allow for testing more complex techniques or a higher amount of options, leading to better and more accurate assessments of the results.

Finally, the study is limited to publicly available data. Although the model proposed accounts for the possibility of including users' metadata, this was not possible to obtain due to privacy policies. For data owners, such as Twitch.tv, this type of data and many others are possible to include, which could lead not only to better results but a better understanding of the model and its overall performance.

5.3 *Implications and Future Research*

The present study shows the relevance of using other types of interactions besides viewing time for live streaming channel recommendations. It is suggested to include this type of data for recommending channels when possible. Nonetheless, further exploration of users' following is needed to properly assess its potential for channel recommendations. Comparing the results of models using different types of interactions or combinations of these is a promising challenge.

The model proposed in this paper produced decent performance in the field of live streaming channels, and it serves as a true baseline score for future studies to come. Still, testing different approaches, such as Deep Learning methods, are in need to accurately assess if the proposed one is the best one to use for this specific field. Nonetheless, it adds evidence to the claims of its capacity to be applied across domains. Furthermore, some research paths deserving of exploration are exposed from the results of this study, such as comparing the performance improvement of adding side information vs. the improvement by hyperparameter optimization of this model, which would be very significant in contexts of limited resources.

6 CONCLUSION

Accurate recommendations can add great business value, especially in fields that have seen a rapid growth in the content they offer, such as live streaming. For this reason, the continuous development and research of different approaches becomes relevant.

This study aimed to implement and analyze the performance of a hybrid latent representation recommender system for live streaming channels based on users' followings and channels' metadata. Considerable efforts for obtaining, processing, understanding and modeling the data were required, but the goal was accomplished successfully. During the analysis, it was found that adding side information in the form of item features, as well as tuning the hyperparameters, does have a positive impact on the performance of the proposed model leading to an optimal version of the model that outperforms other simple baseline models.

This original approach provides information not only for a better understanding of the user's preferences and dynamics for live streaming channels, but also on the performance of RS based on latent representations, and it contributes to a little explored niche academic field, that is, of the recommender systems for live streaming channels.

Aggarwal, C. C. (2016). Content-based recommender systems. In *Recommender systems* (pp. 139–166). Springer.

Babaee, M., Tsoukalas, S., Babaee, M., Rigoll, G., & Datcu, M. (2016). Discriminative nonnegative matrix factorization for dimensionality reduction. *Neurocomputing*, *173*, 212–223.

Barjasteh, I., Forsati, R., Ross, D., Esfahanian, A.-H., & Radha, H. (2016). Cold-start recommendation with provable guarantees: A decoupled approach. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1462–1474.

Bell, R. M., & Koren, Y. (2007). Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, *9*(2), 75–79.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2).

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331–370.

Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y. S., Compton, P., & Mahidadia, A. (2010). Collaborative filtering for people to people recommendation in social networks. In *Australasian joint conference on artificial intelligence* (pp. 476–485).

Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*(6), 1487–1524.

Chen, W., Niu, Z., Zhao, X., & Li, Y. (2014). A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web*, *17*(2), 271–284.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Galuzzi, B. G., Giordani, I., Candelieri, A., Perego, R., & Archetti, F. (2020). Hyperparameter optimization for recommender systems through bayesian optimization. *Computational Management Science*, *17*(4), 495–515.

Gandolfi, E. (2016). To watch or to play, it is in the game: The game culture on twitch. tv among performers, plays and audiences. *Journal of Gaming & Virtual Worlds*, *8*(1), 63–82.

Gunawan, A. A., Suhartono, D., et al. (2019). Music recommender system based on genre using convolutional recurrent neural networks. *Procedia Computer Science*, *157*, 99–109.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, *16*(3), 261–273.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., &

Storkey, A. (2017). Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*.

Ji, Y., Sun, A., Zhang, J., & Li, C. (2020). A re-visit of the popularity baseline in recommender systems. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval* (pp. 1749–1752).

Johnson, M. R., & Woodcock, J. (2019). The impacts of live streaming and twitch. tv on the video game industry. *Media, Culture & Society*, *41*(5), 670–688.

Kar, P., Narasimhan, H., & Jain, P. (2015). Surrogate functions for maximizing precision at the top. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (Vol. 37, pp. 189–198). PMLR.

Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender systems: issues, challenges, and research opportunities. In *Information science and applications (icisa) 2016* (pp. 1179–1189). Springer.

Kleinberg, J. M. (1999). Hubs, authorities, and communities. *ACM computing surveys (CSUR)*, *31*(4es), 5–es.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*.

Lam, X. N., Vu, T., Le, T. D., & Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on ubiquitous information management and communication* (pp. 208–211).

Lin, C.-Y., Chen, T.-S., Chen, J., & Chen, C.-Y. (2021). Personalized live streaming channel recommendation based on most similar neighbors. *Multimedia Tools and Applications*, *80*(13), 19867–19883.

Liu, Y.-W., Lin, C.-Y., & Huang, J.-L. (2015). Live streaming channel recommendation using hits algorithm. In *2015 ieee international conference on consumer electronics-taiwan* (pp. 118–119).

Lops, P., Gemmis, M. d., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, 73–105.

Ma, H., Lyu, M. R., & King, I. (2009). Learning to recommend with trust and distrust relationships. In *Proceedings of the third acm conference on recommender systems* (pp. 189–196).

Maisog, J. M., DeMarco, A. T., Devarajan, K., Young, S., Fogel, P., & Luta, G. (2021). Assessing methods for evaluating the number of components in non-negative matrix factorization. *Mathematics*, *9*(22), 2840.

Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia*

of machine learning, *1*, 829–838.

Miranda, T., Claypool, M., Gokhale, A., Mir, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *In proceedings of acm sigir workshop on recommender systems.*

Newzoo. (2021). *Global esports & live streaming market report.* Retrieved from https://newzoo.com/insights/trend-reports/newzoo-global-esports-live-streaming-market-report-2022-free-version

Oulasvirta, A., Hukkinen, J. P., & Schwartz, B. (2009). When more is less: the paradox of choice in search engine use. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 516–523).

Ouyang, Y., Liu, W., Rong, W., & Xiong, Z. (2014). Autoencoder-based collaborative filtering. In *International conference on neural information processing* (pp. 284–291).

Probst, P., Boulesteix, A.-L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, *20*(1), 1934–1965.

Rana, C., & Jain, S. K. (2012). Building a book recommender system using time based content filtering. *WSEAS Transactions on Computers*, *11*(2), 2224–2872.

Raza, S., & Ding, C. (2019). News recommender system considering temporal dynamics and news taxonomy. In *2019 ieee international conference on big data (big data)* (pp. 920–929).

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web* (pp. 291–324). Springer.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257–297). Springer.

Sharma, R., & Singh, R. (2016). Evolution of recommender systems from ancient times to modern era: a survey. *Indian Journal of Science and Technology*, *9*(20), 1–12.

Shi, Y., Larson, M., & Hanjalic, A. (2010). Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *Proceedings of the workshop on context-aware movie recommendation* (pp. 34–40).

Shi, Y., Larson, M., & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, *47*(1), 1–45.

Shi, Y., Serdyukov, P., Hanjalic, A., & Larson, M. (2011). Personalized landmark recommendation based on geotags from photo sharing sites. In *Proceedings of the international aaai conference on web and social*

media (Vol. 5, pp. 622–625).

Singh, A. P., & Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining* (pp. 650–658).

Sjöblom, M., & Hamari, J. (2017). Why do people watch others play video games? an empirical study on the motivations of twitch users. *Computers in human behavior*, *75*, 985–996.

Smyth, B., & Cotter, P. (2000). A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, *13*(2-3), 53–59.

Thorat, P. B., Goudar, R. M., & Barve, S. (2015). Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, *110*(4), 31–36.

Twitch.tv. (2022). Retrieved 2022-03, from https://twitchadvertising.tv/audience/

Verstrepen, K., Bhaduriy, K., Cule, B., & Goethals, B. (2017). Collaborative filtering for binary, positiveonly data. *ACM SIGKDD Explorations Newsletter*, *19*(1), 1–21.

Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems with Applications*, *158*, 113452.

Weston, J., Bengio, S., & Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-second international joint conference on artificial intelligence.*

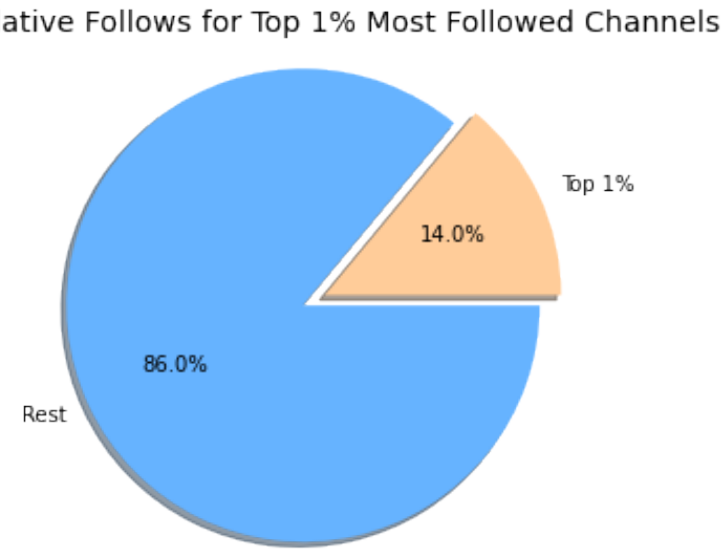
Weston, J., Yee, H., & Weiss, R. J. (2013). Learning to rank recommendations with the k-order statistic loss. In *Proceedings of the 7th acm conference on recommender systems* (pp. 245–248).

Yang, T.-W., Shih, W.-Y., Huang, J.-L., Ting, W.-C., & Liu, P.-C. (2013). A hybrid preference-aware recommendation algorithm for live streaming channels. In *2013 conference on technologies and applications of artificial intelligence* (pp. 188–193).

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, *52*(1), 1–38.

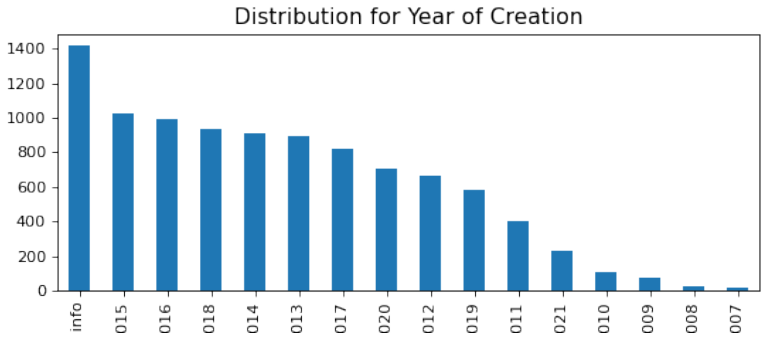
Appendices

A FOLLOWINGS DISTRIBUTION

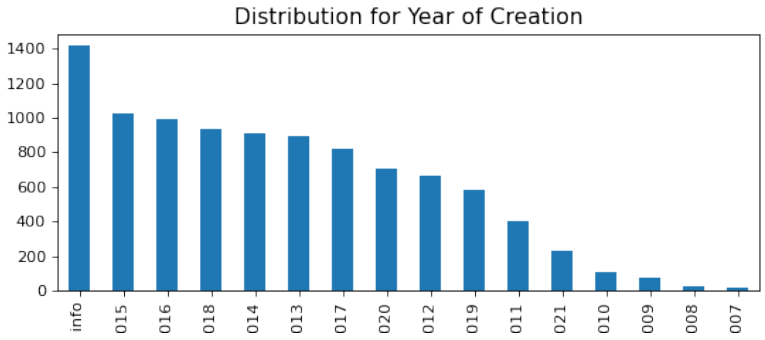


Distribution of Channels Followings

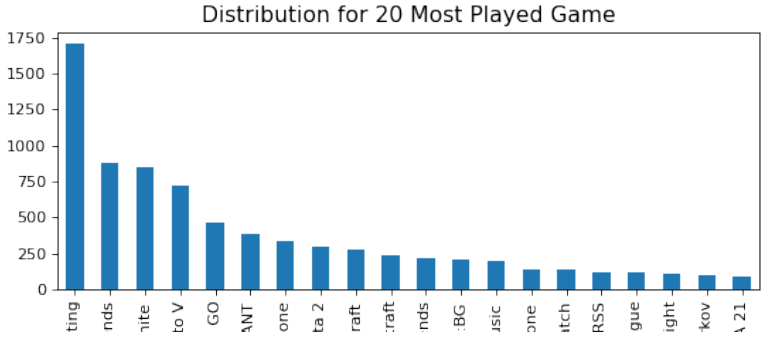
B DISTRIBUTIONS



Distribution of Year Created Feature



Distribution of Language Feature



Distribution of Most Played Games Feature

Table 5: Precision@6 scores for model with selected features, with ten different random configurations of the hyperparameters Number of Components (K), Learning Rate (η) and Number of Epochs.

	Hyperparameters			Score
Iterations	K	η	Epochs	Precision@6
Iteration 1	97	0.123	35	0.246
Iteration 2	24	0.067	47	0.243
Iteration 3	67	0.005	34	0.263
Iteration 4	90	0.023	24	**0.271**
Iteration 5	87	0.016	33	0.264
Iteration 6	14	0.016	36	0.261
Iteration 7	63	0.025	27	0.264
Iteration 8	73	0.035	34	0.256
Iteration 9	25	0.050	25	0.270
Iteration 10	97	0.109	36	0.230
Iteration 11	89	0.022	17	0.259
Iteration 12	81	0.004	11	0.263
Iteration 13	49	0150	45	0.229
Iteration 14	27	0.089	19	0.242
Iteration 15	85	0.082	29	0.237