Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

PICAE – Intelligent Publication of Audiovisual and Editorial Contents

Alexandre Rodríguez Garau

Master in Innovation and Research in Informatics
Data Science

Student: Alexandre Rodríguez Garau,
alexandre.rodriguez.garau@estudiantat.upc.edu
Academic Supervisor: Oscar Romero Moral, oromero@essi.upc.edu
Company: EURECAT
Advisor: Rohit Kumar, rohit.kumar@eurecat.org

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Departament d'Enginyeria de Serveis i Sistemes d'Informació
Facultat d'Informàtica de Barcelona

June, 2021

Abstract

The development in internet infrastructure and technology in last tow decades have given users and retailers the possibility to purchase and sell items online. This has of course broadened the horizons of what products can be offered outside of the traditional trading sense, to the point where virtually any product can be offered.

These massive online markets have had a considerable impact on the habits of consumers, providing them access to a greater variety of products and information on these goods. This variety has made online commerce into a multi-billion dollar industry but it has also put the customer in a position where it is getting increasingly difficult to select the products that best fit their individual needs.

In the same vein, the rise of both availability and the amounts of data that computers have been able to process in the last decades have allowed for many solutions that are computationally expensive to exist, and recommender systems are no exception. These systems are the perfect tools to overcome the information overload problem since they provide automated and personalized suggestions to consumers.

The PICAE project tackles the recommendation problem in the audiovisual sector. The vast amount of audiovisual content that is available nowadays to the user can be overwhelming, which is why recommenders have been increasingly growing in popularity in this sector —Netflix being the biggest example. PICAE seeks to provide insightful and personalized recommendations to users in a public TV setting.

The PICAE project develops new models and analytical tools for recommending audiovisual and editorial content with the aim of improving the user experience, based on their profile and environment, and the level of satisfaction and loyalty. These new tools represent a qualitative improvement in the state of the art of television and editorial content recommendation.

On the other hand, the project also improves the digital consumption index of these contents based on the identification of products that these new forms of consumption demand and how they must be produced, distributed and promoted to respond to the needs of this emerging market.

The main challenge of the PICAE project is to resolve two differentiating aspects with respect to other existing solutions such as: variety and dynamic contents that requires a real-time analysis of the recommendation and the lack of available

information about the user, who in these areas is reluctant to register, making it difficult to identify in multi-device consumption.

The project counts with the participation of the Catalan Audiovisual Media Corporation (CCMA), the University of Barcelona (UB), Eurecat, which coordinates the project through its Big Data & Data Science Unit, the Audiovisual Technologies Unit and the Consultancy Department, Nextret, among others.

This document will explain the contributions made in the development of the project, which can be divided in two: the development of a recommender system that takes into account information of both users and items and a deep analysis of the current metrics used to assess the performance of a recommender system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of the Internet and the increase in information and options available has made personalized recommendation technology increasingly important for consumers, users and also for companies in a wide variety of fields ranging from e-commerce to choose a restaurant. Unlike classic search methods, the PICAE recommender provides the user with a way to access products and services based on the concept of 'discovery', which allows them to access a large number of options in a simple but effective way. For companies, the recommender opens the door to the distribution of their entire catalog, both the most popular and the minority, allows the development of new business strategies and especially in the case before us opens a new relationship with the its users.

The success stories, which are many and well known, indicate the effectiveness of this technology in a wide variety of fields. For example, in the case of e-commerce, it is estimated that Amazon's shopping recommendation system may be generating more than 25% of its revenue, while in the case of the distribution of movies and series, the e-commerce recommendation Netflix could be generating up to 80% of revenue. These differences in effectiveness are not only explained by the differences in the nature of the products, but also by aspects such as the quality of the information handled by the recommender in relation to the catalog of products offered, by a good knowledge and recognition of the user or by the use of a technology and a science sophisticated enough to come to propose a satisfactory 'discovery' to the user.

The PICAE project has the common goal of all partners to develop a recommendation platform that can be used in different but related areas: audiovisual content

(especially TV), textual and news. It is difficult to imagine whether the consumers of the future will be exposed to television, the press and editorial content websites in the way we know them today, but it seems quite clear that access to these platforms must migrate from a model of passive viewer to a model that maximizes and enhances the user's interactive experience at a meeting point between their interests and available content. Recommenders respond to this challenge, although previous experiences in these areas are limited and not easily generalizable. The hypothesis of the project is that the main causes of these partial results are two: the limited modeling of users and the use of low-power tools for proper indexing of content. The project aims to focus on these two aspects and their impact on the engines of recommendation to take the current state of the art further and affect the link between consumers and users. For this thesis we will only focus on the TV use case.

1.1 Partners

Before going deeper into the technical details of the project let us quickly review the most relevant partners:

- **EURECAT**:

  Eurecat is the main technological center in Catalonia. It provides the industrial and business sector with differential technology and advanced knowledge to respond to their needs for innovation and boost their competitiveness. The added value provided by Eurecat accelerates innovation, reduces spending on scientific and technological infrastructures, reduces risks and provides specialized knowledge according to the needs of each company.

  Eurecat's activity is aimed at all business sectors but, in particular, at the 7 strategic areas of the Research and Innovation Strategy for Smart Specialization in Catalonia (RIS3CAT): Food, Energy and resources, Industrial systems, Design-based industries, Industries related to sustainable mobility, Healthcare industries and Cultural and experience-based industries.

- **Coporació Catalana de Mitjans Audiovisuals** (CCMA):

  Public Television and Radio Service in Catalonia. Broadcast on TV3, 33 / Súper3, 324, Esport3, TV3 HD, Catalunya Ràdio, Catalunya Informació and Catalunya Música. Production of audiovisual content. Distribution of content on digital platforms (internet, smartphones and smart TVs). The CCMA occupies a prominent place in the audiovisual and communication sector in Catalonia. In particular, it has an 18% share of television consumption in Catalonia, with TV3 being the leading audience channel in recent years. Catalunya Ràdio also occupies a preferred place among the country's radio stations.

Although in terms of consumption and especially income, access to content through the new platforms still represents a small part of the global, for some years now, the CCMA has placed special emphasis on the development of services within the new distribution platforms (Internet, Mobile Phones, Connected Televisions ...) understanding that audiovisual consumption will gradually move from classic broadcasting platforms to the digital environment. The very change of name of the corporation that went from being the *Corporació Catalana de Ràdio i Televisió* to *Corporació Catalana de Mitjans Audiovisuals* is significant of this vision.

- **Universitat de Barcelona** (UB):

  The UB unit participating in the project is the UB Data Science Group (DATA SCIENCE @ UNIVERSITY OF BARCELONA). DATA SCIENCE @ UNIVERSITAT DE BARCELONA has been set up as a candidate group for TECNIO, a seal awarded by the Generalitat de Catalunya through ACCIÓ. The DATA SCIENCE @ UNIVERSITY OF BARCELONA is made up of various research groups from the UB with the aim of adding synergies to address technology transfer projects in the field of Data Science and the processing of massive data. These groups belong to the Faculty of Mathematics and Computer Science and Physics and are the *Complexity Lab Barcelona*, the Computer Vision and Learning Group and the Computer Vision Group.

- **NexTReT**

  NexTReT optimizes their clients' ICT area with managed services and ICT infrastructure projects, applications and quality of service solutions. Big Data is a concept that has emerged in recent years. Hadoop technologies have given the usual systems of Business Intelligence, analytics and artificial intelligence, the ability to work with very large volumes of data that generate new applications for organizations. NexTReT enables companies, and especially IT departments, to adopt standard Big Data technologies. In this way, a framework based on a new technology is enabled that responds to new demands, but also to the evolution of traditional business data environments. NexTReT is a Partner of the main Big Data technologies in the sector and contributes its experience and knowledge to the construction of innovative solutions, helping clients in public administration, digital media and banking.

1.2 Objectives

Let us take a look at the high level objectives of the project:

1.2.1 High Level Objectives

1. **Improve efficiency and effectiveness in the production and distribution of content**:

   The knowledge acquired by the PICAE recommendation platforms, plus all the knowledge acquired by user profiling techniques are a solid basis on which to improve the distribution of content becoming increasingly personalized and useful to the receiver thus improving the your satisfaction. In the same way that automated and more accurate content indexing will allow us to minimize efforts to classify and archive the ever-increasing number of material both created by broadcasters and by users themselves in an environment of participation and interaction. These two elements will allow accurate knowledge to improve effectiveness and efficiency, but also the success of content production and distribution.

2. **Search for new methodologies of knowledge and interaction with users**:

   PICAE clearly places its core in the search for new methodologies of knowledge and interaction with the audience / users and with the paradigm shift that involves the digitalization of the sector towards a model where user participation is increasingly important and where the term audiences, with a clear passive connotation and which until now had been the driving force, has been replaced by the user, always related to interaction and its ability to decide implicitly or explicitly. Knowing who our users are and want, how, where, with whom they enjoy audiovisual content will allow us to surprise, propose new content that adapts to the profile of the person and the precise moment.

1.2.2 Technical problems to be solved

The PICAE project is articulated in four main axes that mark the development of the new digital economy: (i) to know our users better, (ii) to better classify our contents to provide a better (iii) personalized recommendation service that guides the user in the universe of information, contents and possibilities (iv) in the most automatic, friendly and reliable way possible. But it also adds an axis that makes all this possible: the use of Big Data systems and tools that allow us to process the huge amount of data required by the growing customization of **user profiles** and more accurate **content indexing**, in addition to responding in real time to the needs of the proposed tool.

- **User profiling**:

This project introduces the vision from the field of Complexity Sciences and the so-called Computational Social Science, mixing massive data with knowledge about human behavior in a multidisciplinary way.

1. Introduction of user clustering from the field of complexity sciences and which in some cases have been successful in the field of genetics. The tools would discriminate the most relevant variables in order to be able to characterize various communities or types of users by taking a step further on existing algorithms in the industry with techniques from complex networks and block models that are applied for example in data Twitter.
2. Analysis of cause-effect relationships through new algorithms that go beyond the observation of correlations in the framework of users' interpretation of audiovisual and editorial content.
3. Three-step approach: Stimulus, Outcome, Response from behavioral sciences. Enrichment of the definition of profiling through this approach and through controlled experiments that allow to refine the user profile both in laboratory conditions and in operating platforms (virtual laboratories).
4. The user in his social environment. Behavior in Networks and how it influences their knowledge. Influential users and virality.

- **Indexing**:

  This project proposes to develop and apply textual, auditory and visual information indexing techniques based on deep learning that allow to advance the state of the art in the following lines:

  1. Build a taxonomy of audiovisual concepts that are useful from the point of view of content indexing aimed at the consumption of editorial, news and TV content. This taxonomy must include from purely descriptive concepts of the content (e.g. 'it is a content that speaks of the last elections to the Parliament of Catalonia", 'the actions happens in an urban setting and "La Marseillaise" is playing in the background') to concepts related to the user experience (e.g. 'video has a very dynamic structure in terms of visual and sound'). This task should not cause great difficulties from the point of view of definition, but it should also generate a set of labeled data that allow learning algorithms to build a model. The big challenge is being able to label a large volume of data with minimal effort.
  2. Develop a series of automatic tagging tools that efficiently process large volumes of data according to the above taxonomy. In this case the problem can be divided into 2 subproblems:

(a) Define a deep learning software architecture that is able to learn filters automatically from tagged data. This architecture should take into account the multimodal reality of the data (text, audio, video) and combine the three sources.

(b) Define an operating scheme, at the infrastructure level, of the resulting filters that allows their application to the large volumes of data that are generated daily on a TV.

3. Study the temporal aspects (consumption in sequence of different products) and context (place, time, type of access, etc.) that characterize the consumption of different types of content. Looking at the final recommendation, it is important to know what consumption patterns certain content follows. The information generated would be of the following type: 'Political reports with a high content of interviews are preferably consumed in urban areas on weekend nights from connected televisions (smart TV). These users then view a new episode of a regular series.'

- **Recommendation**:

  While recommendation systems have proven their worth in different application scenarios, such as e-commerce platforms like Amazon, or on-demand video consumption platforms like Netflix, there is no generic solution that can be applied in all the cases, being necessary to contribute specific solutions for each case. In the case of this work, the platform to be developed will have to satisfy the needs of personalization of two different use cases such as the publishing world and an *a la carte* TV platform, where the challenges to be solved will be:

  1. Systems that combine the recommendations obtained by exploiting the new systems of indexing and representation of the content and profiling of users developed in the project, along with more "traditional" ones such as those based on collaborative filters. New hybridization techniques of recommendation systems will be studied such as the application of probabilistic models that combine different algorithms [1], models that train different algorithms together [2], or even and all involve users in choosing the model they like best [3]
  2. Be able to offer recommendations to anonymous users, called session-based referral systems. Unlike applications where the user is registered and historical information is available, in this case it must be customized taking into account only the current session. There are other cases, such as classified ads, where, even if the user is registered, the goal of the user's search is highly dependent on the session. The present work will

evaluate state-of-the-art methods using recurrent neural networks [4] to understand user behavior, as well as strategies that apply NowCasting approaches [5] typical of meteorology applied in this case to content consumption.

3. Generate context-sensitive recommendations. It is a fact that in many cases the incorporation of contextual information into recommendation systems improves quality [6]. In the objective of Picae it is especially relevant because we find cases of use where the social and current context is very relevant (current events, time, ...). This incorporation of the context represents a challenge given that there are different approaches; from the incorporation of contextual factors (location, company, purpose of the search, ...) as a filter before or after generating the list of recommendations, or include them directly in the learning model, or even all the very choice of the most relevant contextual factors for the user [7]. In the project, in addition to the inclusion of context in indexing tasks, the algorithms that incorporate contextual factors into the model such as those based on Factorization Machines [8] or bias-based Context will be investigated. -Aware Matrix Factorization (CAMF) [9][10], among others.

These are the three main technical problems that will need to be solved in order to have a state of the art recommender system. These, however, are really complex and laborious tasks and are designed to be carried out in a very long span of time by different teams of engineers. In this thesis we will focus mainly on the third task.

This will include research about the current state of recommender systems (2), an overview of the main criteria and metrics to evaluate the performance of a recommender system (3), a justification for the selected solution (4), a thorough evaluation of the data at hand (5) and finally a description of the implemented solution and experiments and their results (7).

Chapter 2

Recommender Systems: State of the art

In this chapter we will take a look at the current status of the recommender systems landscape and highlight the most relevant solutions for this project.

2.1 What is a Recommender System?

A recommender system is a programming tool that is used to suggest contents to the users and other entities using a wide variety of strategies. The incremental information overload that the users have been experiencing in the last 25 years led to the development of the first recommender systems. They hoped to help users choose from the huge amount of options that are now available. While traditionally recommendations were based on the recommendations made by experts, now they have become automatically produced by recommender systems. In their beginnings (beginning of 90s) these tools were heavily inspired by the study subject of other closely related research disciplines such as Human Computer Interaction or Information Retrieval. [11]

There are many different recommendation strategies among which we can highlight Content-Based Filtering, Collaborative Filtering, Demographic Filtering or Knowledge-Based Filtering.

2.1.1 Classic Recommender Strategies

The collaborative filtering strategy is based on the basic assumption that user preferences are maintained over time. Likewise, users' own preferences are used to look for similarities between users and make recommendations based on these. This type of recommender mainly suffers from *cold start* problems —new user or new item — and *gray sheep* —users that do not fit in any taste cluster.

Content-based recommenders are based on the basic assumption that people who have liked a particular content will also like other content with similar attributes. The quality of the recommendation is delimited by the content attributes selected from the contents themselves. Although to a lesser extent, they also suffer from the problem of cold start. To a greater degree, however, its main problem is that of overspecialization.

As for the demographic filter recommenders, their use is not very widespread lately, as it involves the use of attributes and characteristics of users that are private. [11]

Finally, knowledge-based recommendations use knowledge of both users and content to infer which content meets user preferences and thus generate [12] recommendations. The KBF subset, known as constraint-based, seeks to be able to recommend complex content that is rarely consumed because it involves limitations on the part of the user —often based on price, e.g. buying a car. [13]

As can be seen, each family of recommenders has a number of strengths as well as a number of limitations. As a result, the first hybrid systems emerged, which we discuss in the next section.

2.2 Hybrid Recommender Systems

Each recommender system has a set of strengths and weaknesses. Thus, while collaborative filtering systems use user community ratings to make recommendations, content-based recommenders use single-user ratings combined with content descriptions to make recommendations. There are other methods, such as knowledge-based ones, that are based on explicit specifications made by the users themselves in order to be able to follow the recommendations.

Knowledge-based systems do not suffer from the problem of cold start, but suffer from not being able to have historical interaction data. Collaborative filter systems provide diversity and serendipity, but suffer greatly from the problem of cold start. This does not affect content-based recommenders to the same extent, but instead they may sin of over-specialization.

Hybrid recommendation systems combine two or more recommendation strategies in order to benefit from their complementary advantages. Most studies combine collaborative filters (CF) with other techniques; often using a weighting between

models [11]. Hybrid recommenders not only provide complementary benefits, they also try to reduce the main problems that affect each of the basic recommenders, such as the problem of cold start and data sparsity.

2.2.1 Taxonomy

There are multiple methods for hybridizing different recommendation systems. In this section we explore the three main types: ensemble design, monolithic design and mixed systems.

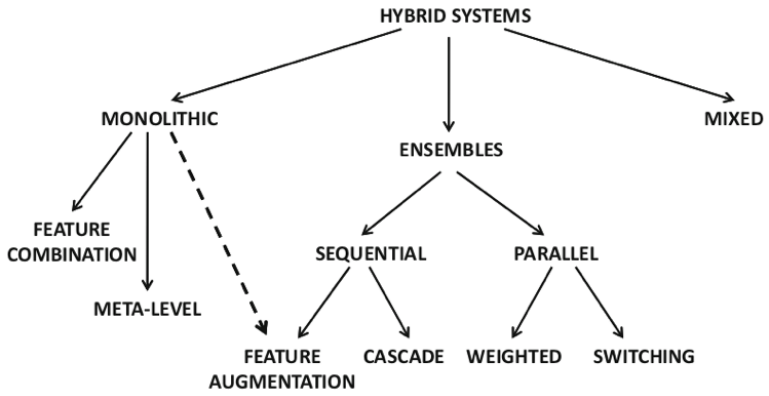

Figure 2.1: Taxonomy of Hybrid recommenders [14]

Hybrid recommender systems are closely related to the field of set analysis in standard classification tasks. In a way, every ensemble is a hybrid system.

Ensemble

This design has as its main characteristics the fact that it directly uses the recommendations of the different recommenders to hybridize, as well as the fact that it produces a unified prediction. Each model in the set produces a result, so the models end up being interchangeable and their output easily combinable. For example, predictions of neighbor-based models could be combined with predictions of latent factors.

The set design seeks to combine the predictions of k recommending systems. Formally, we have k rating matrices, each of them of $(m \times n)$, where m indicates the number of users and n the number of contents. So we want to combine the output of each of these matrices so that we can return a single matrix.

But how is this done? There are two main approaches, the parallel and the sequential.

1. **Parallel**: The parallel design consists of models that operate independently, their outputs combined only at the end. Again, these systems can be subdivided into Weighted and Switching.

- *Weighted*:The weighted models seek to combine the different predictions of the set recommenders, and apply a weighted mean in order to generate the final prediction. The search for optimal weights represents the element of complexity, and can range from a simple uniform weight distribution to other techniques involving heuristics or models. Formally, weighted hybrid systems try to estimate the matrix of R ratings by combining the output of k recommenders. This combination has some weights that we need to estimate: To make this estimate, robust regression models can be used, using the resulting coefficients as the respective weights of each of the models. Also, as mentioned in [15], use Bayesian networks. This model computes the probability distribution over the expected valuation. The weight of each recommendation strategy is automatically selected, adapting the model to the specific conditions of the problem. In the model bag, we can add different types, or the same types with different configurations. For example, we could add models of different types such as collaborative filter engines such as neighborhood based, combined with SVD systems and other types. The idea is to reduce the inherent bias of each of these models. Otherwise, we can use a bag of variants of the same model, such as a collaborative filter model, using different similarity metrics for example.
- *Switching*: This type of design emerged mainly as a possible solution to overcome the problems of cold start. As with the weighted design, it also allows you to operate the various recommendation systems without modifying them, but the use you make of each of them depends on the situation. A classic example case is a combination of a collaborative filter recommender with a content-based one [11]. For a user about whom we have little information, the changing design will make us use the content-based recommender, which tolerates situations near the cold start much better. For a user of whom we have a lot of information, the system can switch and move to use a recommender based on collaborative filter, which can work better in these cases. The changing design does not necessarily involve the use of a different set of models. The same model can also be used, e.g. collaborative filter, with different weights and refinements. In this way we have multiple instances of the same type.

2. **Sequential**: As the name suggests, sequential design requires that the use of the recommenders that make up the system be done sequentially, i.e. in order. Thus, the output of one model can feed the input of the next. We have two methods for hybridizing recommenders sequentially: the cascading method and the feature augmentation method.

- *Cascade*: The first recommender of the set makes a series of recommendations, which are evaluated by the next model of the queue, which tries to improve them. This sequence continues until we reach the last model of the set. This technique is known as *boosting* in the field of classification. [14]
- *Feature Augmentation*: In this type of hybrid design, the output of one of the models is used as the input of the next without any modification. If modifications are needed, then we are talking about a monolithic design.

Monolithic

Monolithic design involves creating an integrated recommendation algorithm using various types of data. The main result of this type of design is that the various data sources are more strongly integrated, making it difficult to evaluate the components of the recommendation as 'coming from different black boxes' [14].

There are two main methods of hybridization within monolithic design: the combination of features and meta-level systems.

1. **Feature Combination**: We follow this approach when we want to combine heterogeneous data sources in order to achieve a unified representation that can be used in a single recommendation model. Additional steps are often required to achieve this; either by modifications of the algorithms to be used, or by means of an additional preprocessing of the input data.

2. **Meta-level**: Just as in the feature combination system the data is combined together, in the meta-level systems, it is the models themselves that are combined. A good example is when they combine collaborative filtering systems with content-based systems. Thus, in order for the collaborative filter recommender to be able to use the characteristics of the contents, it must be modified. For example, special weights can be added for the purpose of optimizing the latent factor [14].

Mixed

These systems involve the use of multiple recommendation algorithms, but the contents recommended by the various systems are presented separately. For example, an online store may have several recommendations displayed in different sections of the same page; one recommends more popular trends, another recommends based on historical purchases, and so on.

Chapter 3

Recommendation Metrics

The purpose of this section is to determine the most appropriate metrics for evaluating the performance of the CCMA recommendation system. An initial introduction to the most relevant concepts when evaluating the effectiveness of recommendations is provided and a general review of the most used metrics in this field is continued. It explains the events and information that must be recorded in order to be able to use each of the metrics introduced and addresses the different ethical challenges that these types of systems can generate.

3.1 Recommender System Evaluation

Since the advent of digital commerce and VOD platforms, recommendation systems have proven to be of great help to users in the face of the paradox of choice. The large extension of the catalogs of these platforms, usually generates in the user a strong indecision on which product to choose. The more alternatives you have, the more dissatisfied you feel with the decision you make. That is why recommending systems are a great help to the user, as they facilitate the decision-making process by reducing their choice of space.

In addition, they can be a great advantage for the business. These systems can generate personalized recommendations, i.e. the user is offered products based on their preferences, which causes niche products to be shown only to the appropriate audience, greatly increasing the sales / views of these type of items, at the same time that a high satisfaction of the user is obtained. There are many possible strategies

that can be used when building a recommendation system: collaborative filtering, factoring machines, deep learning techniques, etc. But regardless of the approach we choose, the same question always arises: **how good is our recommender?**

Assessing how good the recommendations are is as or more important than the architecture we use to build the recommendation. That is why it is important to define a methodology that allows us to evaluate these recommendations.

3.1.1 Recommender evaluation concepts

In the context of the recommendations, the fact that most concerns researchers and professionals in this field is the satisfaction of the end user. Under this pretext, what is being tried to achieve with the recommendations is that they have a substantial value for the user and that he can make the most of them. In other words, we don't want our recommender to always suggest the same songs, always show products in the same category, or never suggest new content to view. That is why we need to explore and define the different concepts to consider when evaluating the recommendations of our system.

Utility

Utility, also known as the relevance or satisfaction of the recommendation, is equated with the order of preference in consumption. It is strongly related to the user's consumer interest and therefore to their preferences and tastes.

Novelty

The concept of novelty refers to the idea of having elements in the recommendations that have not been previously recommended to the user. In this context, we can divide the concept into 3 different levels: novelty at the user level, novelty at the system level and novelty at the recommendation level.

- Novelty at the **user level**, or level 1, is understood as any element (included or not in the system catalog) that the user did not know until the moment of the recommendation.

  Defining metrics to assess this level of novelty is by no means trivial, as information from outside the system should be considered in order to measure what the user knows or does not know.

- Novelty at **system level**, also known as a Level 2 Novelty. It refers to system catalog items that have never been recommended to the user. To assess this level of innovation in the recommendations, it is necessary to know the history of the user within the platform.

- Novelty at the **recommendation level**. The third level of novelty considers only the elements within each recommendation. Under this pretext, novelty is defined as elements within the recommendation that have not been repeated. Unlike the previous two levels, the metrics that evaluate this type of novelty do not require any prior information about the user.

Diversity

Within the recommendation systems, diversity is explained as the variety of elements within the recommendations. This concept is very important when evaluating our recommender, as, depending on the purpose of our system, recommendations with low diversity may be of little interest to the user.

Unexpectedness

In the context of recommendation systems, user expectations are defined as the elements that the user would like to consume. Thus, an unexpected recommendation would consist of deviating from these expectations, avoiding obvious or uninteresting recommendations, with the possibility of surprising the user.

Serendipity

A serendipity is defined as a chance or unforeseen discovery made by a researcher in the course of a goal-oriented research with different theoretical assumptions. Serendipities occur unplanned and occur unexpectedly. In the field of recommendation systems, serendipity is defined as an innovative and useful recommendation for the user, but at the same time surprising and unexpected. If we look at it, the concepts of utility, novelty and unexpectedness are included in serendipity, but it should not be confused with any of them. An unexpected recommendation may surprise the user, but it does not have to be new or useful to him / her. The same thing happens with utility and novelty.

Coverage

Coverage refers to the extent to which a recommender system can make recommendations. The concept can be divided into two branches: catalog coverage and user coverage.

1. **Catalog**: The coverage of a catalog (or items) refers to the percentage of catalog items that the recommender can recommend. In this sense, a far-reaching recommender corresponds to a system that only considers a small part of the catalog available to make recommendations.

2. **User**: User coverage is defined as the proportion of users to whom the recommender system can make recommendations. The concept can be a bit confusing, as a priori a recommender system should be able to make recommendations to any user, but if we think about the cold start problem the user reach gains strength relevance. In this situation, although the recommender can make predictions for the user, his / her confidence in the recommendation is very low.

3.2 Metrics

In this section we set out the most relevant metrics when measuring the concepts defined in the previous section.

Symbol	Meaning
R_u	List of recommendations for users u.
U	Set of Users
I	Set of Items.
H_u	Consumption history of user u.
C_u	Items consumed by user u.

Table 3.1: Symbols and their meaning in the field of Recommender Systems

3.2.1 Utility

Metrics for measuring the usefulness of recommendations focus on how the user values or reacts to recommendations made by the recommender. They tell us if the recommender has captured the interests and tastes of users, but they totally overlook such important concepts as diversity or novelty. We classify them into three different types.

Error-Based

Error-based metrics are widely used to determine the accuracy of predictions. In a system where we can assign a score, $r(i)$, to the elements that users have viewed, either by inference or by explicit assessment of users, these metrics tell us how wrong our system is when it comes to correctly guessing the rating, $p(i)$, that a user will give on a given item.

$$\text{util}\,(R_u) = MAE = \frac{\sum_{i \in R_u} p(i) - r(i)}{|R_u|}$$

Mean Absolute Error (MAE) and Root Mean Squared Error ($RMSE$) are two of the most used metrics in this context.

Classification Based

Two of the most common metrics in the literature are precision and recall. Accuracy, P, is defined as the proportion of items within the recommendation, R, that the user has consumed. On the other hand, the coverage, r, answers the following question: *of all the items consumed, C, which have been recommended?*

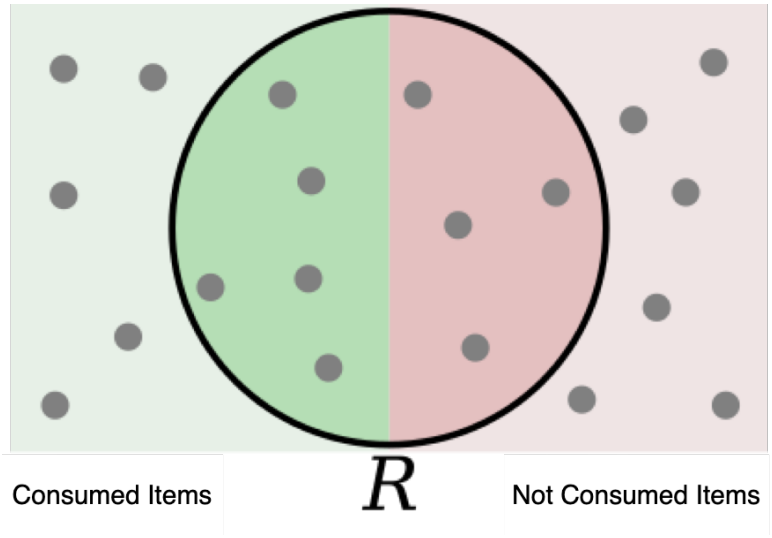

Figure 3.1: Diagram depicting the recommendation possibilities

$$\text{util}\left(R_u\right) = P@N = \frac{\left|C_u \cap R_u^{(N)}\right|}{\left|R_u^{[N}\right|}$$

$$\text{util}\left(R_u\right) = r@N = \frac{\left|C_u \cap R_u^{(N)}\right|}{|C_u|}$$

Where $@N$ determines up to what position within the recommendation, $R_u^{(N)}$, we want to measure the metric. For example, if the recommender system recommends 9 items, the accuracy and coverage will not be the same if we consider only the first 4 items than if we consider the entire list.

Note, however, that while accuracy and coverage are good indicators when evaluating the utility of recommendations, they completely overlook the order of their elements. In this case, metrics such as Average Precision (AP), Normalized Discount Cumulative Gain ($nDCG$) or *R-score* come into play.

Average accuracy is characterized by being a metric that indicates how good our system is when it comes to sorting recommendations by relevance:

$$\text{util}\left(R_u\right) = AP@N = \frac{1}{\min\left(N, |H_u|\right)} \sum_{k=1}^{N} \text{rel}\left(i_k\right) \cdot P@k$$

Where, in this case, $\mathrm{rel}\,(\mathrm{i}_k)$ represents whether the element is relevant, 1, or not, 0, by the user. But in terms of relevance, we don't need to stick to a binary classification. The elements can have different degrees of relevance depending on the user's preferences. In this case, it is useful to consider the normalized discount cumulative gain:

$$\mathrm{util}\,(R_u) = \frac{DCG_p}{IDCG_p}$$

Where,

$$DCG_p = \sum_{k=1}^{p} \frac{\mathrm{rel}\,(i_k)}{\log_2(k+1)}$$

And $IDCG_p$ is the ideal DCG_p, which is what we would get from a perfectly sorted recommendation by relevance. This metric is often used in applications where the list of recommendations is very long and the user is expected to view many of the recommended items. Therefore, the penalty suffered by each element decays logarithmically with its position.

In other systems, however, the number of items that the user sees before choosing one is very small, so the penalty in these cases must fall faster. The *R-score* metric applies precisely this principle:

$$\mathrm{util}\,(R_u) = \sum_{k=1}^{|R|} \frac{\max\,(r\,(i_k) - d, 0)}{2^{\omega_k}}; \omega_k = \frac{k-1}{\alpha - 1}$$

Where $r\,(i_k)$ is the user's rating on the element i_k, d represents the 'neutral' rating, and α is the parameter that controls the exponential decrease. For example, in a system where the user can score up to 5 contents, $r\,(i_k)$ can take values between 0 and 5, and id would be equivalent to the score 3. The metric would only take into account those elements of the recommendation with a valuation greater than 3, and its value would decrease exponentially depending on its position, k.

Finally and probably most importantly, in cases where the recommendations list is not really meant to be that long the metric that is most commonly used in recommender systems is *Hit Rate*.

As with *NDCG*, Hit rate can be defined at various levels depending of how many top elements of the recommended list we want to consider. It works in a really similar way: instead of looking in which position the relevant items ranked it just considers weather the items in the list are relevant.

One way of calculating this is by taking all the items that users have rated —in our case all the videos that a user has watched—, take out the last one as ground truth (Leave-One-Out cross-validation). Then train the model and, for each user, check if the ground truth is present in the top-k recommended items for that user.

```
def HR(train_set, test_set):

    hit_ratio_5 = 0
    hit_ratio_10 = 0
    hit_ratio_25 = 0

    n_users = len(train_set["user"].unique())
    counter = 0
    for user in list(train_set["user"].unique()):
        counter +=1
        #generate 25 top recommendations for current user
        recs = get_user_recs(model, interactions, train_set,
            nrec_items=25, show=False)
        #get ground truth for current user
        item = list(test_set[test_set["user"] == user]["item"])[0]
        # calculate hit rate
        hit_ratio_5 += hr_at_k(user_list=recs, test_item=item, k
            =5)   #@5
        hit_ratio_10 += hr_at_k(user_list=recs, test_item=item, k
            =10) #@10
        hit_ratio_25 += hr_at_k(user_list=recs, test_item=item, k
            =25) #@25

    hit_ratio_5 /= n_users
    hit_ratio_10 /= n_users
    hit_ratio_25 /= n_users

    print("HR@5: {:.2f}%".format(hit_ratio_5*100))
    print("HR@10: {:.2f}%".format(hit_ratio_10*100))
    print("HR@25: {:.2f}%".format(hit_ratio_25*100))
```

Listing 3.1: HR method

Where the method `hr_at_k` is simply:

```
def hr_at_k(user_list, test_item, k):
    return 1 if test_item in user_list[:k] else 0
```

Listing 3.2: HR method

We do that for all users, sum all the times that the ground truth has been found in the top-k and then divide by the number of users. This way we obtain the Hit Rate or Hit Ratio of this recommender.

Online Evaluation

The utility of the recommendations can also be assessed through online experiments. That is, to measure how good the recommender is once it is already making predictions in a real work environment.

The best known metric in this type of experiment is *click through rate* (CTR). This metric measures the proportion of recommended items that the user has interacted with. It's a way of assessing how successful the recommender has been in capturing the user's attention.

$$\text{util}\,(R_u) = CTR = \frac{|C_u|}{|R_u|}$$

On the other hand we have the visit after recommended rate (VRR). This metric captures the fact that a user interacts with content that had been previously recommended. In other words, at the time of the recommendation, due to its context, the user may not or does not want to see the recommended items, but if they are relevant to him / her and consume them later. Last but not least, retention. Retention measures the effect that the recommendation system has on users continuing to consume items or use the platform. This metric is usually evaluated using A / B tests where the difference between the retention times of two control groups is measured.

3.2.2 Novelty

In this context, the metrics attempt to determine how innovative the recommender system is in making recommendations. It is sought that the recommender proposes elements that have never been shown to the user. It should be noted, however, that it is not evaluated how useful these recommendations are for the user, which can lead to a low level of confidence on their part in the recommendations made by our system. We will focus on the recommender system level novelty and mention recommendation level.

System Level

One of the most used metrics at this level of novelty is based on the similarity between the recommendation elements and the elements in the user history[16].

$$\text{nov}\,(R_u) = \sum_{i \in R_u} \min_{j \in H_u} d(i,j)$$

Where, $d(i,j)$, is a distance function that measures the similarity between the elements i and j. Then, the novelty, nov, of the recommendation by the user, R_u, is represented as the sum of $min_j d(i,j)$ on the elements, i, of R_u. The more similar the

elements of the recommendation are to the elements of the user's history, the lower the value of the metric, which translates into a low novelty in the recommendations.

Other metrics [17][18] consider the popularity of the elements in order to determine the degree of innovation they bring to the user:

$$\text{nov}\,(R_u) = \sum_{i \in R_u} \frac{\log_2 pop(i)}{|R_u|}$$

$$\text{nov}\,(R_u) = 1 - \frac{|pop(i)|}{|U|}$$

Where popularity, $pop(i)$, is calculated based on the number of users who have consumed the item, $|R_u|$ and $|U|$ represent the number of items in the recommendation and the number of users in the system respectively.

Recommendation Level

In order to measure the novelty at the recommendation level, in [19] it is proposed to take into account the classification of the elements within the recommendation and apply a reduction factor, $disc(k)$, due to having to explore the list. In addition, the metric also considers the probability that the user has seen the element, $p(seen|ik)$.

$$\text{nov}\,(R_u) = \sum_{k=1}^{|R_u|} \text{disc}(k)\,(1 - p\,(\text{seen} \mid i_k))$$

3.2.3 Diversity

The most widely used metric in the literature is known as *intra-list similarity.*[20]:

$$\text{div}\,(R_u) = \sum_{i \in R_u} \sum_{j \in R_u} d(i,j); \quad \forall i \neq j$$

Where the function, $d(i,j)$, is the function responsible for measuring the similarity between the elements i and j. Cosine similarity is commonly used, although any function that defines the distance between each pair of elements in a set can be valid.

3.2.4 Unexpectedness

The simplest metric that can describe how unexpected the recommendations of our model are is the proportion of unexpected elements it contains[17]:

$$\text{unexp}\,(R_u) = \frac{|R_u - EX_u|}{|R_u|}$$

Where EX_u refers to user expectations, and $|R_u - EX_u|$ indicates the number of Ru elements not included in EX_u. The main problem with this metric is how

to define user expectations. Another metric introduced in [21], uses the point-wise mutual information (PMI) function to evaluate the concept:

$$\text{unexp}\left(R_u\right) = \sum_{i \in R_u} \sum_{j \in H_u} NPMI(i,j)$$

Where NPMI, is the standardized PMI:

$$\text{NPMI}(i,j) = \frac{\text{PMI}(i,j)}{-\log_2 p(i,j)}$$

$$\text{PMI}(i,j) = \log_2 \frac{p(i,j)}{p(i)p(j)}$$

$p(i)$ is the probability that the element i will be scored or evaluated by any user and $p(i,j)$ the probability that both elements will be evaluated by the same user. $NPMI$ can take values between –1 and 1. At 0, it would tell us that the evaluation of i and j are completely independent, -1 that will never occur together and 1 that have total co-occurrence. That is, to surprise the user, what we are looking for is for this metric to be as close as possible to 0 or for it to take negative values.

3.2.5 Serendipity

Although in the literature we find a strong consonance in its definition, the concept of serendipity is quite complex. Therefore, it is not entirely certain that the metrics proposed by the different authors will be able to quantify the serendipity of the recommendations. In the case of [22], the metrics they use take into account the concepts of utility and surprise:

$$\text{ser}\left(R_u\right) = \sum_{k=1}^{|R_u|} \max\left(R_u[k] - EX_u[k], \quad 0\right) \text{rel}\left(i_k\right) \frac{\text{count}_k(k)}{k}$$

Using the user's expectations, they evaluate the surprise that each element brings to the user. In addition, they take into account their relevance and position in the recommendation. In [23], they simplify the previous metric, so that they do not consider the order of the elements in the recommendation:

$$\text{ser}\left(R_u\right) = \sum_{i \in UNEXP_u} \frac{\text{rel}(i)}{|R_u|}$$

Where $UNEXP_u = R_u - EX_u$, and $rel(i)$ represents whether the element i is useful (1) or not (0) for the user. Finally, in [24] they use the similarity of the cosine to measure the similarity between the elements of the recommendations and the user's history:

$$\text{ser}\,(R_u) = \frac{1}{|H_u|} \sum_{i \in H_u} \sum_{j \in R_u} \frac{\text{cossim}(i,j)}{|R_u|}$$

In this metric, low values indicate that the recommendations deviate from the user's usual behavior and, therefore, that they are innovative for the user.

3.2.6 Coverage

The simplest metric that can evaluate the coverage of the catalog is the one proposed in [23], where they define the catalog as the proportion of items that can be recommended over the total size of the catalog:

$$\text{cov} = \frac{|I_p|}{|I|}$$

On $|I|$ represents the total number of items in the catalog, and $|I_p|$ the total number of items the recommender system can recommend.

Also, in [23] an alternative metric is proposed, where what is measured is the range in a given space of time:

$$\text{cov} \ = \frac{\left|\cup_{j=1.N} I_L^j\right|}{|I|}$$

Here, N indicates the number of recommendations observed in a certain space of time.

Alternatively, in [18] they use the Gini coefficient:

$$\text{cov} \ = \frac{1}{|I|-1} \sum_{j=1}^{|I|} (2j - |I| - 1) p\,(I_j)$$

This metric takes values between 0 and 1, where 0 indicates that all elements have the same probability of being recommended and 1 would correspond to the system always recommending the same element.

3.3 Trackers

In the previous sections we have reviewed the different concepts and metrics to consider when evaluating the performance of the recommendation system we implement. But, as we've seen, depending on the metric or metrics we choose, we'll need some kind of a priori information about users and their interactions with the platform.

First of all, the most essential aspect in almost all metrics is the history of the user within the platform. Knowing what content the user has viewed, H_u, is critical to determining their tastes, preferences, and expectations. And not only that, any

interaction a user may have with an item: clicks, "view synopsis", "add to my list", ratings, etc.

Second, but not least, is the information that tells us whether or not the user has come to see the list of recommendations. That is, the platform must capture the user's movement through the application. For example, on a VOD platform, you know where the recommender is located, if the user moves within the recommender, what items in the list of recommendations have been displayed on the screen.

Chapter 4

Hybrid Design

This section proposes an initial design of the hybridization engine and justifies why. First, a description is made of the sequential recommenders, a type of advanced model based on deep learning that has been considered for the design of the hybridization engine. The exploration carried out on the consumption data is continued, in order to be able to estimate the real possibilities of integrating a sequential recommender into the hybridization engine. The section continues with an explanation of a recommended system such as factoring machines, and explains its suitability for the use of the PICAE project.

4.1 Considerations for sequential recommendation

The main goal of traditional recommendation systems is to model users' preferences for different content based on the implicit or explicit interactions that occur between users and elements. However, they assume that all user-content interactions in the sequence history are equally important, leading to the user having widespread preferences over time. Also, not taking into account the sequential dependencies between user interactions leads them to an inaccurate modeling of their preferences.

User behavior not only depends on their long-term preferences, but also largely depends on their current intention, which could probably be inferred and / or influenced by more recent interactions. Just as we can enter a music portal one day looking for a 'quiet session' and another day we look for something more moving, also when other content is consumed, e.g. television, there are times when we look

for comedies, and others when we may love more less insubstantial content. It is in this approach that the use of sequential recommenders as possible candidates for entering a hybridization engine is proposed for the PICAE project.

This subsection consists of two parts. An explanation of the types of sequential recommendations as well as the case that applies to the PICAE project, followed by a study of the consumption data made to evaluate the available sequences themselves and evaluate the feasibility of such recommendations.

4.1.1 Sequential recommenders

Sequential recommenders, in addition to capturing users' long-term patterns, also seek to model the relationship between the different actions the user takes within a single session.

These types of recommenders are based on deep learning techniques. Specifically, what are known as recurrent neural networks —RNN.

There are a number of basics that we need to keep in mind when we talk about sequential recommenders. They are as follows:

- **Object of the behavior**. Refers to the items, services or content with which the user interacts. It can be associated with other relevant information such as descriptions, images or interaction time.
- **Types of behavior**. It is defined as the way the user relates to the object of the behavior. For example: search, click, buy, add to cart, share, and more.
- **Behavior**. Element-element combination. For example, in a digital commerce, a user who buys a smartphone (interaction, element).
- **Sequence (or session)**. For a single user, it is defined as the combination of multiple behaviors.
- **Sequential recommendation system**. System that takes as input the different sequences of a user and recommends the most appropriate elements, services or content using recommendation algorithms.

Considering these aspects, the next point to assess is the type of the sequence itself. There are several, which we summarize below:

- **Sequences based on experience**s. Sequences in which the user interacts with the same element multiple times. Take for example the case of a user who enters a digital business: first search for words in the product search engine, then select the product that interests you, then click on the details section (technical specifications, sizes ...) of the product, add it to the shopping cart and finally the purchase.

- **Transaction-based sequences**. Behavior sequences where the user interacts with different elements, but the type of interaction is fixed - for example, buying. The most characteristic example would be a digital business that only records the purchases of its users.
- **Mixed sequences**. It's a mix of sequences based on experiences and transactions. These are the sequences that best fit a user's behavior on digital platforms.

Finally, and depending on the type of sequence available, sequential recommenders have the following objectives:

- **Based on experiences**. In experience-based sequential recommenders, the user interacts with the same item multiple times. The purpose of such recommendations is to predict the next interaction that the user will have with the item in question.
- **Based on transactions**. In transactional-based sequential recommenders there is only one type of behavior. The purpose of such referrals is to recommend the next item that the user will purchase given their transaction history.
- **Based on mixed sequences**. Recommenders based on mixed sequences are the most complex: each sequence is made up of different types of behaviors and multiple objects of behavior. These types of recommenders are expected to be able to model the sequential dependencies between different interactions, different elements, and different behaviors.

The present case of the PICAE project places us in the framework of a sequential recommender based on transactions. This is because there are multiple elements with which users interact —content— but this is done through a single type of transaction —consumer event. That is, initially there are no clicks, or other types of events, but the interactions are determined by the *a posteriori* event in which a user has consumed for a certain number of seconds, a certain content.

In order to be able to consider the use of this recommender within the hybridization engine, it is necessary to evaluate the available sequences with the consumption data.

4.1.2 Data Evaluation

The data that we have had access to during the development of the project has been changing constantly, mainly due to confidentiality concerns. In the first stages of the project the only source of data that we had access to was a sample contained in a `.xlsx` file. Later on, the CCMA dumped a week's worth of data in a Hive

table that we could take advantage of more easily through Spark. Finally, halfway through April we had access to the whole history of consumption data since 2019.

The initial work on consumption data was carried out on the small sample of just over ten thousand records, contained in the file `Big Data_ESBD-130 Relació entre conceptes de consolidació.xlsx`.

Once the concept of sequence is defined, i.e. number of transaction events within a session for a user —either registered or anonymous— we now need to characterize it. In order to be able to see which sequences could be available as a starting point, a study was made of the available consumption data.

Below is a summary of the sample:

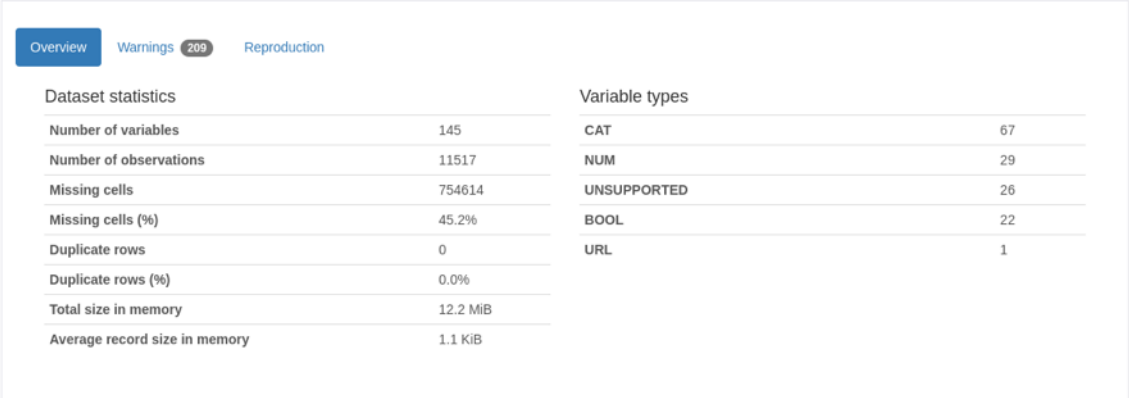

Figure 4.1: Description of the initial content sample

Thus, we observe how a total of almost 150 columns were available, half of which contained categorical information. It is important to note that 45% of the sample was empty, suggesting a possible redundancy in the actually usable data. In fact, much of the data is constant and, as such, does not provide discriminatory information:

Figure 4.2: Constant variables in the sample

Thus, we appreciate how the information coming from the columns `a.n_sn`, `a.bo_canal_sense_arafem` ... does not add value and, consequently, is removed

from the study. There are a series of highly correlated variables and, above all, a series of variables that are virtually copies of each other, becoming redundant. This is the case for columns like `st_user_id, st_player_id` and `max_an`. These columns were omitted from the study. With respect to the null values, in spite of representing almost half of the available cells, they have high concentrations in some variables:

Figure 4.3: Null values in the sample

In the image above we can see a sample of columns where, with the exception of the column `a.sn`, the rest of the columns are virtually empty, which excludes them from the study.

4.1.3 Transformations

The steps conducted to have a working dataset look as follows.

1. Column prefix removal
2. Removal of empty and / or highly sparse columns
3. Removal of duplicate columns
4. Elimination of invariant columns

The final set of information can be seen in the following table (4.1):

Col Name	Description	Type
an	whether the user can be tracked or not	user
bo_es_diferit	pre-recorded or live	user
min_data	timestamp when the user started watching	user
bi_segons_consum	total seconds spend watching	user
usuari_id	user/device identifier	user
player_id	random value to identify the session of the user	user
font	TDT or IP	user
media_tipus	video or audio	content
producte_id	HbbTV, CCMA web platform...	user
contingut_id	Content identifier	content
sbt	indicates whether the subtitles are active	user
canal_nom	channel name	user
programa_id	program identifier	content
programa_nom	name of the program	content
durada	length of the content	content
programa_capitol	episode of the program	content
media_titol	name of the content	content
tematica	thematic of the content	content
dispositiu_model	model of the device	user
dispositiu_vendor	manufacturer of the device	user
dispositiu_browser_nom	name of the browser	user
dispositiu_tipus	kind of device (tablet, PC, phone, smartTV)	user

Table 4.1: Selected Columns

In essence, these columns contain the main information that we can have through the use of content consumption. It is information highly consistent with the sources studied by the team in charge of the user profiling. We end with a set similar to the following:

min_data	bi_segons_consum	usuari_id	player_id	an	font	media_tipus	producte_id
2019-07-12 15:56:42	440	0010f680-7e3c-11e9-933c-fd2e60408420	274301562939772307	False	IP	video	HbbTV_tv3
2019-07-12 22:20:43	581	008e9820-5e15-11e9-94bc-efe7da616b9f	762171562962831302	False	IP	video	HbbTV_tv3
2019-07-12 16:45:19	604	00a38a30-af12-11e8-9d7e-21076e1f4100	725931562942659835	False	IP	video	HbbTV_tv3
2019-07-12 00:00:43	985	00b424c0-475c-11e9-8657-39f2acd60f34	537971562881325054	False	IP	video	HbbTV_tv3
2019-07-12 00:00:17	1540	00b44610-f18c-11e8-84ed-e376327a38da	98491562881354076	False	IP	video	HbbTV_tv3
2019-07-12 19:04:26	1905	023dd9a0-83c3-11e9-b498-ddaf0e324120	851631562951003500	False	IP	video	HbbTV_tv3
2019-07-12 20:03:46	1909	02527e90-9f58-11e9-862a-fb60b4cac567	73511562954594276	False	IP	video	HbbTV_tv3

Figure 4.4: A small sample of the cleaned data

4.1.4 Sequences

The above work serves two purposes. The first, an initial familiarization with consumption data and the second, to see what sequences are available.

The first problem with sequences is that, strictly speaking, there are none in the initial sample analyzed. That is, for each user and session, only one consumer event is available. In other words, we have sequences of an event, which makes them uninteresting. It is for this reason that it was decided to explore a larger sample in size.

Before we had access to the whole data —which was late due to the signings of all partners of NDAs being delayed— the CCMA provided us acces to a HIVE database through Apache Thrift.

The Hive sample consists of approximately 300,000 records. Although, in comparative terms, it represents a minimal fraction of the consumption data available at the CCMA, if we assumed that this sample is representative of the whole we would obtain the following:

- More than a third of the sessions are uni-event
- More than 50% of the sessions are of two events
- 5% of sessions have three or more events

This does not make it absolutely unfeasible to propose a sequential model based on transactions, but it does limit its possible usefulness.

While it is true that the concept of sequence itself could be changed in order to adjust it to the reality of the data, e.g. considering sequences of events over several days, in a first iteration **it is ruled out to hybridize sequential models for recommendations**.

4.2 Monolithic Design

As we saw earlier in the section on hybridization taxonomies, there are many techniques that can be used. Depending on the technique, we also need to know, not just how we will hybridize but what we will hybridize.

In the PICAE project, and in view of the preliminary results of the consumption data, it is temporarily ruled out making use of transactional-based sequential recommenders. This leaves us with other options to combine. For example, collaborative filter systems, content-based, knowledge-based systems, and so on.

To make the most of the data available, as well as the profiling and indexing exercise, we need a recommender that allows us to use it. That is why we introduce the concept of factorization machines as a monolithic system that provides an excellent format for conveying all this data and incorporating it together. Thus, not

only can we capture user interactions with content, but we can also add features that result from user profiling and content indexing.

We then dedicate a section to factorization models, to then introduce factorization machines as a generalization of these models, highlighting their benefits and possible application to the context of the PICAE project.

As a previous step to introducing factorization machines we will first explain latent factors.

4.2.1 Latent Factors

Latent factor models are one of the most technically advanced models currently within the recommendation systems landscape. These models take advantage of dimensionality reduction techniques to fill in empty entries - ratings.

The idea of dimensional reduction is to rotate the axis of the system to eliminate correlations between dimensions. The key to these methods lies in reduced, rotated, and fully specified representations, which can be robustly estimated from an incomplete data matrix. Once this specification has been obtained they can be rotated back to the original system axis.

The use of these correlations is critical for all collaborative filter methods, whether neighborhood-based or model-based. For example, user neighborhood-based models employ correlations between users, while item-based ones employ the correlation between them. Matrix factorization methods provide a way to take advantage of all the correlations between rows and columns at once in order to estimate the entire matrix[14].

Latent factor models address collaborative filtering with the holistic goal of discovering the latent factors that explain the observed ratings of the contents. Examples of techniques that exploit latent factors include pLSA, Neural Networks, and models that are based on user-content valuation matrix factorization — based on Singular Value Decomposition (SVD). Recently matrix factorization models have gained a lot of popularity thanks to the accuracy and scalability they offer.

Factorization models are recommendation systems that belong to the collaborative filtering branch. The main idea of these algorithms is the decomposition of the matrix of interaction between users and items in the matrix product of two rectangular matrices of remarkably reduced dimensionality.

Matrix factorization-based models map both users and items into a space of latent factors of dimensionality f, such that user-item interactions are defined as scalar products within that space.

The latent space attempts to explain ratings by characterizing both items and users with factors automatically inferred from user feedback. If the items are movies the latent factors can be as simple as comedy vs. drama, amount of action or orientation to a child audience, more complex factors such as the development of

the characters or the "peculiarity" of the film and even completely uninterpretable factors or dimensions.

4.2.2 Matrix Factorization

Given an matrix of ratings A, where m is the number of users and n is the number of items, the model learns:

- A Matrix of latent factors $U \in R^{m \times d}$ where the row i are the latent factors for the user i, where d is the number of latent factors.
- A Matrix of latent factors $V \in R^{n \times d}$ where row j are the latent factors for item j.

Figure 4.5: A simple example depicting matrix factorization

Latent factors are learned such that the product UV^t is a good approximation of the matrix of ratings A. In the previous image 4.5 we can see how the matrix A (on the left) can be expressed as the product of U (4×2 left of the resulting matrix) and V (2×5 above the resulting matrix). It is important to note that the input (i, j) of UV^t is simply the scalar product of the vectors $\boldsymbol{U}_i$ and $\boldsymbol{V}_j$ (latent factors) of the user i and the item j, which we want to be as close as possible to the value $A_{i,j}$. It is important to see that matrix factorization typically provides a more compact representation than learning the entire matrix. This would have $O(nm)$ inputs or positions while the latent factor matrices U and V have $((n+m)d)$ positions, where d is usually much smaller than m and n. This is why matrix factorization finds a latent structure in the data, assuming that the observations are close to a low-dimensional sub-space. In the above example this advantage is not seen due to the size of the matrix, but in real systems the factorization of matrices can be much more compact than the whole matrix.

How do the matrices U and V look like?

We will first define a function that will be the one we will try to optimize when we make the UV^t product. Intuitive function would be square distance. That is, we will minimize the sum of the squares of the errors on all observed pairs of ratings.

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2$$

It should be noted that in this formula we only consider observed pairs (i, j), i.e. values other than 0. This is not a good idea as an matrix of 1s would minimize the error and produce a model that does not make good recommendations and that does not generalize. This corresponds to the first case on the left of the following figure 4.6

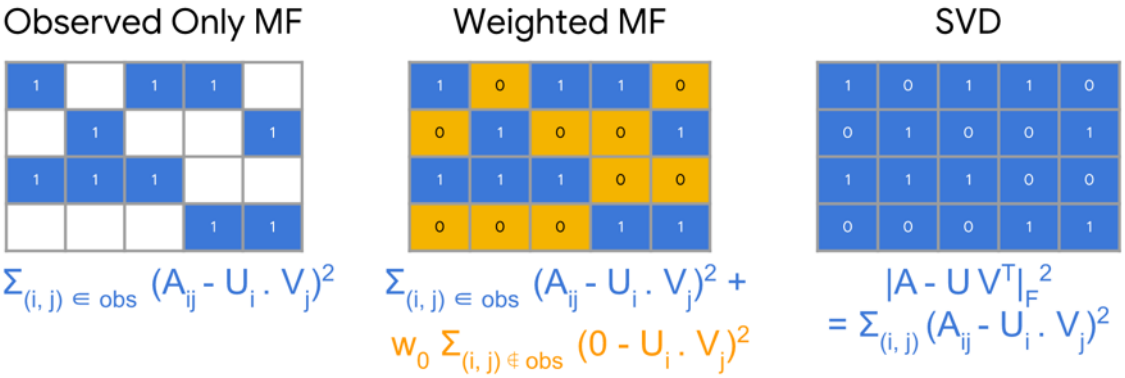

Figure 4.6: Objective function optimization

Alternatively, unobserved values such as 0 can be treated and all entries in the matrix added together. This is called the Frobenius distance between A and its UV^t approximation. This would correspond to the second case in the image 4.6.

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \left\| A - UV^T \right\|_F^2$$

This problem can be solved by SVD, but it is not the best solution because in real systems the matrix A will be very sparsely populated (sparse, videos seen by a user as opposed to all videos on the platform) and therefore the UV^t solution it will be very close to 0, thus giving a very poor generalization. This solution is represented on the right of the figure 4.6.

For this reason, weighted matrix factorization is proposed, which consists of 2 terms:

- Sum of the terms observed.
- Sum of unobserved terms (treated as zeros)

$$\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{ obs}} (A_{ij} - \langle U_i, V_j \rangle)^2 + w_0 \sum_{(i,j) \notin \text{ obs}} (\langle U_i, V_j \rangle)^2$$

w_0 is a hyperparameter that gives weight to the terms so that the target function is not dominated by one or the other. In the training process fine-tuning this

parameter is key. In real systems it will also be important to assign a weight to the observed entries to eliminate the disproportionate weight that very popular content may have. These can also dominate the target function. The objective function, therefore, will be:

$$\sum_{(i,j)\in \text{obs}} w_{i,j} \left(A_{i,j} - \langle U_i, V_j \rangle\right)^2 + w_0 \sum_{i,j \notin \text{obs}} \langle U_i, V_j \rangle^2$$

where w_{ij} is a function of the frequency of the user i and the item j. Minimization of the objective function. There are several algorithms to minimize the objective function, among which we highlight:

- Stochatic Gradient Descent: Generic method of target function minimization
- Weigted Alternating Least Squares (WALS): method designed specifically for this particular goal.

The complexity is quadratic for both matrices U and V. The operation of WALS is as follows: Initially random values are given latent factors and then alternate between

- Fix U and solve V and
- Fix V and solve U.

Each iteration can be solved exactly (using a linear system) and can be distributed. This method ensures convergence as each step ensures that the target function will decrease[25].

In our particular case, we are interested in WALS in particular because it has a working implementation in PySpark's MLLib and allows us to process the massive amount of data that involves making individual recommendations for all users of the platform.

4.2.3 Factorization Machines

In principle factorization machines (FM) work in a relatively similar way to Matrix factorization although they are a more general concept. Factorization machines appear as an evolution of Support Vector Machines (SVM). SVMs do not work well with sparse data and this issue is solved by FMs by adding latent factors. On the other hand, FMs also try to solve specific problems that come from Matrix Factorization models. We mainly talk about two specific problems:

1. Matrix Factorization models are not applicable with standard data prediction vectors (i.e. a feature vector within the R^n space) and

2. Matrix Factorization models are usually derived individually for each specific task and require modeling and design of learning algorithms.

FMs are more general predictors that are able to estimate parameters even under high sparsity conditions. Defined relatively formally, FMs have a standard prediction task in which they try to estimate a function that, given a vector $x \in R^n$ returns a value T (where $T = R$ in a case of regression, $T = +, -$ for classification e.g.). It is assumed that we have a training set D of the form $D = (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ...$ where we have feature vectors and function values that we are trying to estimate.

So far we have only had one matrix with user identifiers and items, along with those observed interactions. In the factorization models we did not have the possibility to use information about users or items or to express the training data set in the form of data vectors. In FM, however, we do have these possibilities and we must express the observed rating as a set of interactions that detail the user, the item and the rating assigned to the item.

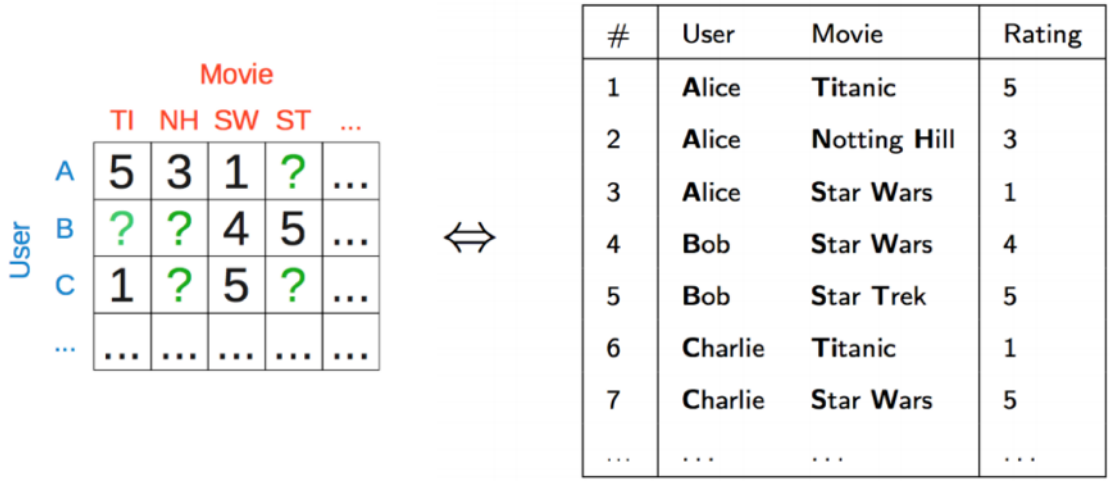

Figure 4.7: Interactions codification

Once this has been done, as in 4.7, this list should be transformed into a set of feature vectors where each user and item will be represented using dummy variables, i.e. a binary indicator that is at 1 if the user who has in fact the valuation is the one represented by that variable and at 0 otherwise. We will do the same for the items. This means that we will have a total of $|U|+|I|$ dummy variables to represent users and items, and in each vector there will be only two values set to 1 in these variables, one for the user and one for the item. This can be seen in the following figure 4.8.

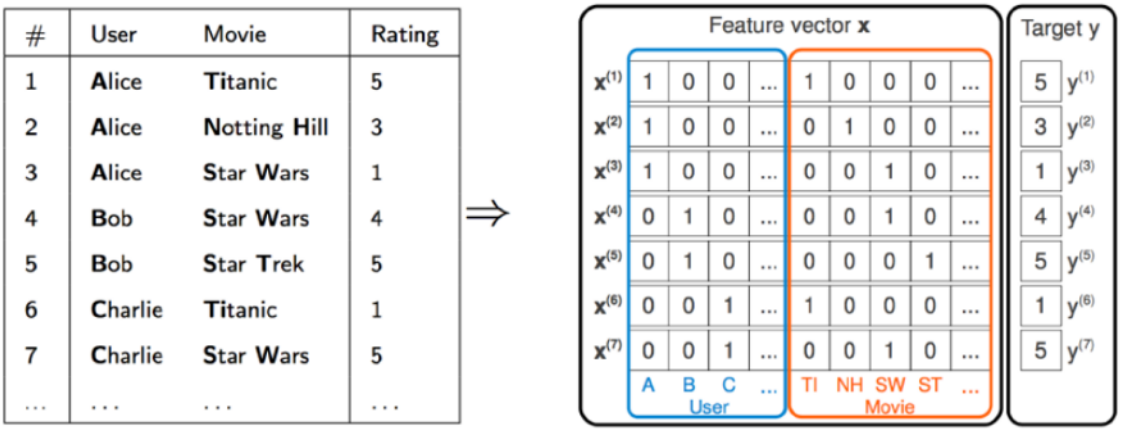

Figure 4.8: Interactions to feature vectors using dummy variables

Finally, we can add other implicit variables to this table that serve to better explain the interactions between user and item, such as display time, other displayed items, content genre, etc. Figure 4.9 shows how this is done.

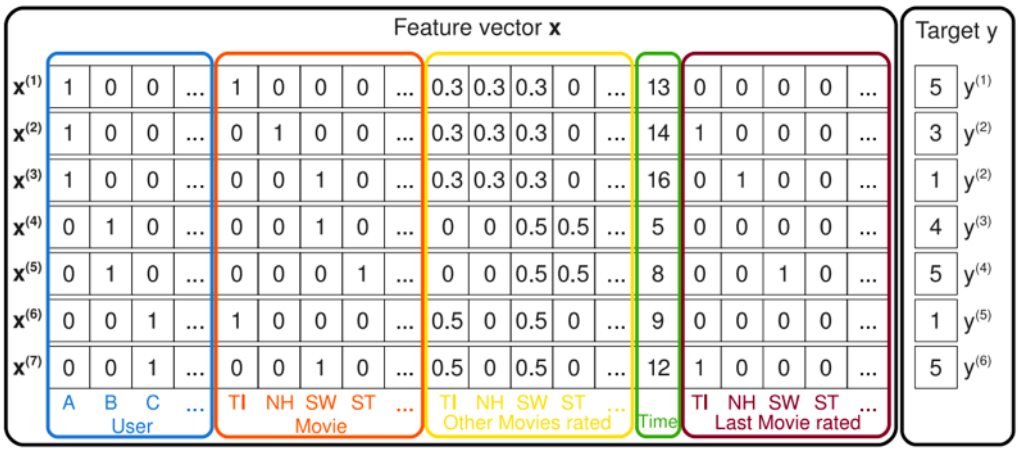

Figure 4.9: Adding the features to the vectors

Given this representation of the data it could be considered a linear regression model to estimate user ratings, but this option is quickly discarded as linear regression is unable to account for interactions between variables (user-item). The solution of a polynomial regression model also does not work for us because the number of dummy variables we have introduced means that in the vast majority of cases $P \gg N$, that is, the number of parameters is considerably higher than the number of observed cases (Curse of Dimentionality).

This is why a method is needed to capture interactions between pairs of variables while also being able to make accurate predictions with very sparse data. The FM is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i x_j$$

Where the parameters to be estimated are $w_0 \in R$, $w \in R^n$ and $V \in R^{n \times k}$. A row $\boldsymbol{v}_i$of V describes the i-th variable with a number of factors $k \in N^+$. k defines the dimensionality of factorization.

This FM is of degree 2 ($d = 2$) and is capable of capturing all interactions between pairs of variables.

The parameter w_0 represents the total bias. w_i indicates the weight or force of the variable i. Finally, the scalar product between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ models the interaction between the variables i and j. That is, instead of using a $w_{i,j}$ parameter for each interaction, FMs model the interactions by factoring them.

4.2.4 Conclusion

Given the above, it is expected that the following can be entered at the input of the factoring machine:

- Contents: feature vector resulting from audio processing
- Contents: vector of characteristics resulting from video processing
- Contents: vector of characteristics resulting from the processing of the text
- Users: vector of characteristics resulting from profiling

User feature and content vectors can, by the way factoring machines operate, enter the model directly without any prior modification. It should also be noted that these vectors can be summarized into a single embedding vector.

Thus, we would have the following:

1. **Computation of interactions**. As shown in figure 4.8, the 'Rating' would be the result of a user's interaction with a particular content. For the case of PICAE, we are assuming an initial binary case, which materializes once 60% of the content is consumed. We could also assume that the fact that a user clicked on the video means it is a positive interaction, meaning that all interactions would have a rating of 1.

2. **Entry of user and content representative vectors**. As seen in figure 4.9, these vectors represent users, taking advantage of the user profiling results; and also represent the contents, thus taking advantage of the item profiling.

In the following chapters we will further explore the data and detail the implementations of the models.

Chapter 5

Data Platform Context

In this chapter we will discuss the data that is available and that will be used to create a working recommender system. Overall, as explained before, to build a recommender only three things are strictly required:

1. User IDs
2. Item IDs
3. Ratings

With these three we can create triplets of information of the form `(user_id, item_id, rating)` with which we can represent the interactions between the users and the items that are available. That being said, let us describe how the data that we have available is organized and has change over the course of the development of the project.

5.1 Data Sources

There are two main data sources and each one of them tell us different information about the main actors in a recommender: the Users and the Items. This sources are Consumption data and Content data. The following subsections will detail how this data is stored, what it contains and why it will be useful to build a recommender system.

5.2 Content Data

This is the source that, to this point, has been exploited the least. Content data contains information about every video that the CCMA has stored in their video repository. This data includes simple information such as the channel where the program was broadcast, the length of the program or the timestamp when it was broadcasted.

However, here we can also find other metadata that can provide information about the content that will be useful when it comes to characterizing the contents that will be recommended. Some examples could be the thematic of the video —Current events, entertainment, fiction, sports, etc.—, the language of the video or the ethical code for that content —for all audiences, 7+, 13+, 18+.

This data would be of key importance if we were trying to create a content based recommender system since it thoroughly describes the items that need to be recommended to the users. Even though this is not the case, this information will come in very handy to create a matrix factorization recommender because we will be able to cross the ids of the videos that have been played with this information and use it as item features.

As of the moment of writing this thesis, the total amount of different programs is 135112. In order to store this data the CCMA decided to store it to an *ElasticSearch* index so that we can have a systems that allows fast response times and also supports many requests. This index is hosted on a NexTReT machine.

We can access this data via python or even URI through the IP where it is hosted, the name of the index and the query. For example, we can count the number of documents in the index:

`http://10.210.4.141:9200/picaecataleg/_count/`


```
{
  "count": 135112,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  }
}
```


5.3 Consumption Data

The other big part of the necessary data that is needed in order to understand and explain the habits of the users corresponds to the consumption data. This is a

considerably bigger data source compared to the content data because it describes every interaction between a user and a content that is either being broadcast or being consumed on on-demand platforms —such as TV3 a la carta, SmartTV apps or mobile apps.

The volume of data is immensely bigger because of the rate at which each dataset grows:

- **Content Data** only increases whenever a new content is added to the CCMA video repository —it doesn't happen too many times a day— which means that this ElasticSearch index will grow very slowly, if at all, because some contents can be taken out of the repository due to various reasons —rights expiration or no longer recommendable.
- **Consumption Data**, on the other hand, does not depend on how much content is being produced, but on how many users are watching the contents at each moment. This means that, the more users there are, the faster the consumption data grows.

We've explained why this source is considerably bigger, but we haven't detailed what this data represents, so we will proceed to explain that below.

Currently, this table contains 1.700 million records and keeps growing by the day. These records contain all the relevant information about the interactions —consumption— between a viewer and a any kind of content in video form belonging to a CCMA channel or platform. Shortly we will explain in which situations a new record is inserted to the table. The table is formed by 145 columns that store all kinds of metadata, ranging from the `user_id`, `seconds_consumed` or `content_id` to `thematic` or `device_vendor`, among many others.

5.3.1 Users

It is important to emphasize the meaning of being logged in. The CCMA has a common online platform for all of the channels —TV3, TV3HD, TV3CAT, 33, 324, Esport3 and Super3— and everybody can watch any content that has been broadcast, as long as they still have the rights. A user can create an account in the platform and access the usual functionalities that a video-on-demand platform. This, however, means that that user will now have a unique id every time they watch content while being logged in.

This is relevant because when the user that is consuming content is not logged in, the data of that interaction is tied to the cookie stored in that device, meaning that if a user that is not logged in watches content in two different devices —a tablet and a TV, for example— two different ids will be stored, corresponding to the respective cookies of each device. The consequence of this is that it is impossible to link these

two different ids to the single person, and when it comes to computing personalized recommendations, these will be calculated for each cookie, and therefore, for each device. It is also possible that the cookies from a device are deleted and a new cookie is created, virtually creating a new user, but this is not very common.

The CCMA has a very open policy about its contents and lets any user, logged or not, access all of the content available. This means that there are not many incentives for users to create an account in the website, to the point where only a 2% of all recorded interactions come from logged users.

There are three cases in which a new record is added to the table:

1. When a logged in user watches content using either the web —`https://www.ccma.cat/`—, the mobile app on either phone or tablet or a TV that is compatible with HbbTV —most TV after 2012 are. In these cases the same user id will appear in all different interactions regardless of the device.

2. When a not logged in user watches content in all the previous cases and the devices accept cookies. Each device will have its own user id because the cookie is not the same between devices.

3. When someone watching TV that does not accept cookies puts on a CCMA channel. In this case the user cannot be tracked using a cookie or a permanent id but the interaction is still recorded and the id field is populated with an anonymous id. Every time the user puts a CCMA channel while channel-zapping a new record is inserted into the table.

These are the three possible scenarios in which new consumption data can be created. This way, an average of 2.071.033 interactions are recorded every day since 01/01/2019.

Overall we can see two different kinds of users: Identifiable users, either via id or cookie, and anonymous users. In a similar vein, the content can also be separated in two different categories: live and pre-recorded —*diferit*. Live content can be consumed through TV and via application. Most pre-recorded content that airs on TV can later be consumed in the platform.

The CCMA separates the two ways of consuming content considering the source of the video. If it has been consumed by watching TV, then this is considered `TDT`. All other methods are considered `IP`. Let us see how the consumption is distributed in the following table.

In this table we can see all the possible combinations considering the source of the content, whether the user is anonymous and the kind of content —live or pre-recorded. It is apparent that there are some missing combinations:

- Content consumed via `TDT` can only be live.

Consumption distribution				
Source	**Anonymous**	**Live**	**Count**	**Percentage**
IP	false	directe	138817370	7.8%
IP	false	diferit	171838816	9.7%
TDT	false	directe	861089000	48.3%
TDT	true	directe	610628359	34.2%

Table 5.1: Interactions grouped by source and live/pre-recorded

- There are no anonymous users for content consumed via `IP`. This makes sense since there is always the possibility of storing a cookie in the device.

Looking at the table we can see that 34.2% of the content is consumed by users that can not be tracked. This means that the information that is generated by these users is not useful, because it does not allow us to track users individually therefore making it impossible to create personalized recommendations.

5.3.2 Live Content

Live content also carries its own problems. A content that is being broadcast might not always be recommendable for various factors.

When a content is being broadcast it might be being recorded live. If that is the case then it means that the video can not be already available on the platform. Because of this, this interaction record cannot reference a content because it does not yet exist in the video repository and thus, the `content_id` will be `null`. This happens for both `TDT` and `IP` sources.

Sometimes, due to rights ownership, the content might no be uploaded at all to the platform or it could be uploaded in small clips —for comedy sketches, for example.

The consequence of this is that even though we have the ability to track many user's consumption, it is almost impossible to link the content that the user watched live on TV to a video in the CCMA repository. There are methods for some of the contents to link them to existing videos in the repository using the production number but this falls outside of the scope of this project.

Summing up, we can conclude that most, if not all, live data is unusable for us at this stage of the project. This accounts for 90.3% of the total data or 1.610.534.729 rows. Therefore only 9.7% of the data —171.838.816 rows— will actually be useful: it uniquely identifies a user —or device— and the content that has been watched.

5.3.3 Model Data

We have carefully analyzed and identified which subset of the data will actually be useful. This dataset will be used to train a recommender that outputs personalized

recommendations for all users. However, we have to take into account some factors.

- To make sure we don't run into a cold-start situation we need to select those users that have a minimum amount of interactions.
- Having multiple devices that aren't logged in may introduce many more users that have few interactions —because every different device will have its own `user_id`.

The current dataset contains a total of 171.838.816 interactions made by 25.787.952 users. Also, the number of different items or videos made by these users is 485.759. As we can see the number of users is very high compared to the total number of interactions and this is because, as mentioned before, every device is considered as a different user. This dilutes the interactions of a single user into many different `user_id` and, unless the user was logged on, it is impossible to link all these interactions to a single user.

The `id`s used to identify the devices are `UUID`s and are generated every time a new cookie is created for a device. They have a really low chance of being duplicated (finding a duplicate within 103 trillion version-4 UUIDs is one in a billion) [26]. Version 4 —the version used— are created randomly whereas the previous versions are created using other terms such as MAC address or timestamps. They are represented as 32 hexadecimal digits displayed in five groups separated by hyphens (8-4-4-4-12).

Contrary to the `user_id`, the identifier `ui` serves as an id that is common for all the interactions made from devices that are logged on with the same account.

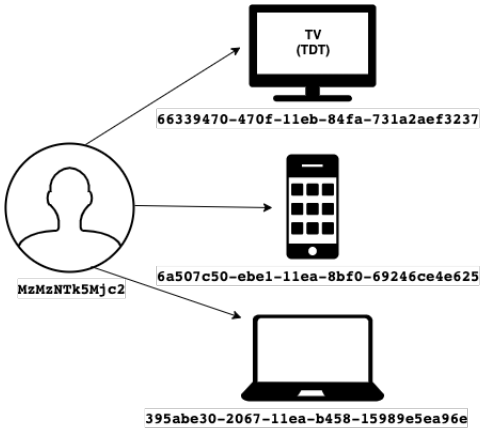

Figure 5.1: Diagram depicting interactions between a logged on user and multiple devices

Furthermore, even if we still wanted to recommend for all different `user_id`, the average amount of interactions per user is 6.75, meaning that there will be many users that will have 1 or 2 interactions. Let us see it in the following figure.

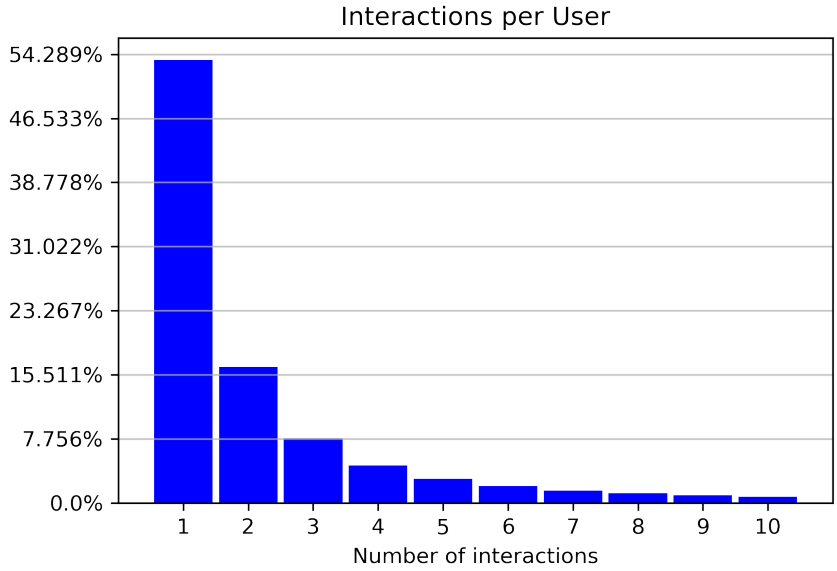

Figure 5.2: Histogram showing the frequency of user interactions (pre-recorded)

	Interactions
count	25787952
mean	6.75
std	4.946
min	1
25%	1
50%	1
75%	3
max	18166

With this short descriptive table combined with the previous histogram we can see how more than 50% of users have only 1 interaction and 77.83% of them have 3 or less. The user with the most interactions had 18166 and, as mentioned before, there are a total of more than 25 million different `user_id`. As expected, this dataset will probably not be good for training a reliable recommender due to the lack of interactions per user.

Table 5.2: Pre-recorded interactions

It is clear that we need to find a more reliable set of users, where the interactions are less spread among users. To achieve that the logical course of action is to use the interactions made only by logged on users. These, as explained before, are identifiable by a platform id even if they are made using different devices. The following table showcases an example of a user that used different devices but can be grouped using the unique identifier `ui`.

ui	usuari_id	segons	cont_id	prog_id	durada	prog_nom	tematica
MzMz...	66339...	5561	6094841	6012425	03:36:06	Tot es mou	ACTUALITAT
MzMz...	6a507...	12427	6067483	6012425	03:40:37	Tot es mou	ACTUALITAT
MzMz...	28666...	113	5853746	6012425	00:04:04	Tot es mou	ACTUALITAT
MzMz...	28666...	9037	5853792	6012425	03:08:39	Tot es mou	ACTUALITAT
MzMz...	395ab...	1490	6038830	4000472	00:24:25	Notícies 3/24	ACTUALITAT
...	...	...	...	...	...	...	...

Table 5.3: Example of a user's consumption with different devices

This identifier only has a value different than null for those users that registered and watched a content while being logged on. We can clearly see in this example that the user has the same `ui` in all interactions but the `user_id` takes up to 4 different values. It is because we have the ability to link the interactions to the same user that this dataset will prove more useful.

Let us check the differences with the previous dataset and a new dataset that only contains interactions of logged on users. First we will start by taking a look at the histogram of number of interactions for this new dataset.

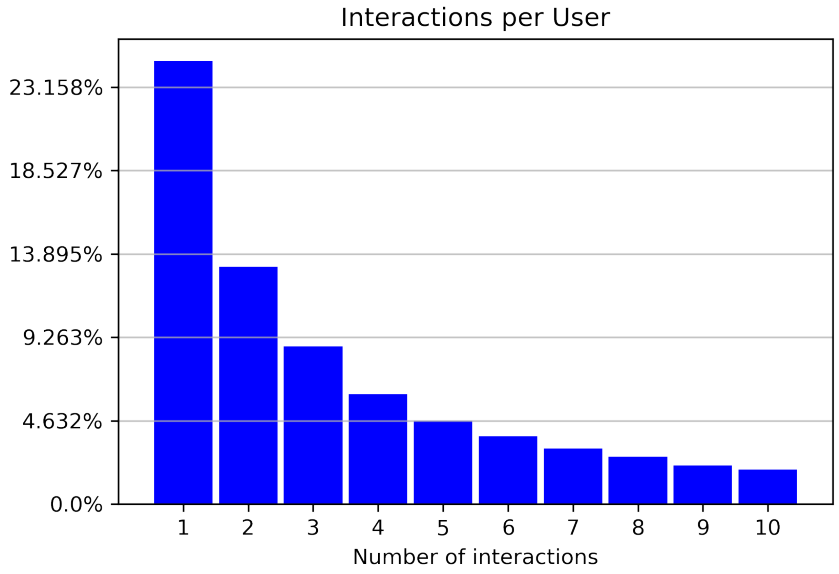

Figure 5.3: Histogram showing the frequency of user interactions (pre-recorded and logged on users)

	count
count	107952
mean	24.13
std	92.752
min	1
25%	2
50%	4
75%	13
max	6896

The difference between the two sets of data is immediately recognizable just by looking at the percentage of users that only have 1 interaction. While in the previous one more than 50% of the users had 1 interaction, this one does not even reach 25%. What is more, 52.63% of the users have 4 or more interactions. Grouping the interactions this way also has made the average number of interactions per user rise from 6.75 to 24.13.

Table 5.4: Pre-recorded and logged on interactions

It is important to highlight that the proportion of interactions with pre-recorded content made by users that are logged on is really small compared to the total of pre-recorded interactions —about a 2%. In numbers, all pre-recorded interactions round the 170 million while logged on pre-recorded are only $\approx$ 2.6 million. However we have to consider two things:

1. The users that are being taking into account in this dataset are probably the ones that are going to benefit the most from a recommender system because they are the most frequent users.

2. A smaller dataset does not mean not significant. There is still a lot of data to process and create a robust recommender.

To sum up, the subset of data that we will end up working with is that which contains all interactions made with pre-recorded content by users that are logged on. It has a total of 2.6 million rows, 107.000 users and an average of 24 interactions per user.

Chapter 6

Ethics in Recommendation

In this short section we will discuss some topics related to ethical recommendation and factors that we have to take into account when producing personalized suggestions to users of the platform. These considerations are specially important in our case because we are building a recommender for a public TV channel.

The recommender has to make *good* recommendations from the point of view of all platform stakeholders, which rises the question: Who are the stakeholders?

One could argue that the stakeholders are the CCMA, the users, the advertisers and even the content providers. These are all the parts that can be impacted by a recommendation. A recommender will have a negative impact if either of the stakeholders gets a negative effect from it or if the recommender violates some rights.

6.1 Usual Recommender Problems

Recommender systems can run into a variety of problems derived from the content that they recommend. Let us quickly go over them:

1. **Recommending inappropriate content**: Depending on the setting there might be some content that is not meant to be consumed by all users.
2. **Privacy**: Since we are using data from individual users about their consumption habits, we are at risk of leaking data that is meant to be private.
3. **Opacity**: Sometimes recommender systems are just seen as black boxes and offer absolutely no explanation for the recommendations that they make.

4. **Fairness and other biases**: Recommender systems can have a discriminatory behaviour when dealing with gender, race, age, etc. and might have other undesired biases.

5. **Transparency and Social Responsibility**: The objective of the recommender must be very clear and we have to avoid *echo chamber* effects, specially for political content.

In our particular case not all problems apply. The vast majority of content is suitable for all audiences. Also, we do not have to worry about discriminatory content because the CCMA has really neutral content regarding these topics. The same applies for *echo chamber* effects.

6.2 Considerations for Picae

While we established that some of the more general problems do not apply, there are some that do, so let us review them and propose a solution.

- **Privacy**:

  We have to include a generic explanation that informs the user about *what* information is being stored and used, *how* it is being processed and *which* protective measures are being applied in order to protect the privacy of the user.

  Also, depending on the level of decision that we want to give the user we can even let them choose whether their data is used by the recommender or not.

- **Opacity**:

  The best way to avoid opacity is to give good and concise explanations about why a content has been recommended to a user. The following figure summarizes different possibilities of doing so (6.1):

Figure 6.1: Different types of explanation for users [27]

While either of these possibilities would be ideal, we have to keep into account that due to the nature of factorization machines it is really complicated to explain why a recommendation is made. Deeper analysis of the subject is required. This is more applicable for item/user based collaborative filterings.

- **Unwanted Biases**:

  One of the most widespread flaws in recommender systems is that they become biased really fast. For example, if we watch a news section and then watch a current events program, the recommender might only recommend content related to current events, obscuring other thematics.

  To avoid these behaviours a solution could be to provide the user with the possibility to select preferences of content that they want/do not want to watch. An alternative way to tackle this issue could be to target diversity as a metric to improve.

  Another issue that is abundant in recommender systems is the popularity bias. We should avoid having the most popular items being the most recommended.

- **Transparency**:

The objective of the recommender is very important to define because it will dictate the course of action to follow once decisions must be taken. In other companies such as Netflix it is obvious that their recommender is key in their business. They want users to consume as much content as possible to keep the engaged, this way retaining them and earning more money. This is probably not applicable in our case because the CCMA is a publicly funded company and does not have the same objectives as a privately managed company.

For this reason, the objective should be clearly defined and explained to the user in a section of the web page.

Chapter 7

Implementation

In this chapter we will discuss the chosen implementation of the Recommender system. As detailed in previous chapters, the recommendation techniques that we decided to explore, aside from the discarded sequential recommenders, are matrix factorization and factorization machines. They each offer a wide variety of libraries that implement these models and in the following sections we will detail the chosen ones, a justification, some details of the implementation and the results. First, however, we will briefly describe the development environment.

7.1 Development Environment and Data

As mentioned in previous sections (4.1.2), the data that has been available has been changing. First we only had access to a `.xlsx` file, then we could access a more significant Hive database and finally to the whole data. This data is updated daily and is stored in a huge `parquet` file in a server provided by NexTReT. Since many people will be working with this data, NexTReT also provided a Jupyter Notebook for everyone as well as enough memory and storage to run processes there.

7.1.1 Livy, Spark and Sparkmagic

Of course, to work with such volumes of data, some big-data processing engine is required. This is why, in order to have a notebook that allows multiple users working at the same time with the possibility of also connecting to a Spark cluster (hosted by NexTReT as well) we needed Apache Livy.

Livy is a service that enables easy interaction with a Spark cluster over a REST interface. It enables easy submission of Spark jobs or snippets of Spark code, synchronous or asynchronous result retrieval, as well as Spark Context management.

To connect, however, Jupyter and the Spark cluster through livy we need a set of tools called *sparkMagic*. Also, it automatically creates a `Spark session` and a `Hive Context` to query the dataframes. The following diagram summarizes the connection:

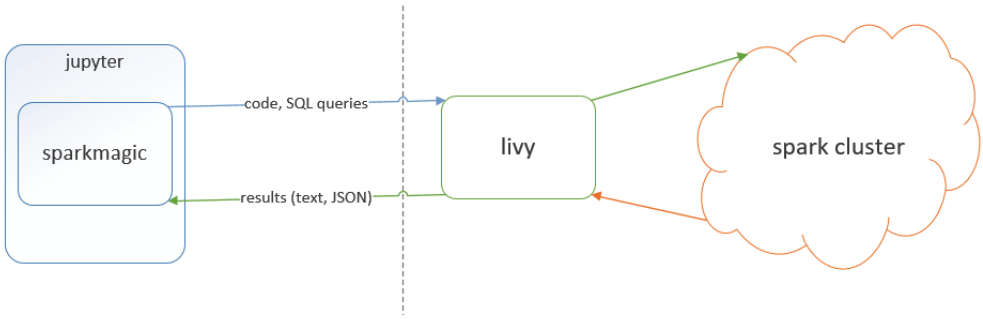

Figure 7.1: Diagram depicting the interaction among Jupyter, Livy and a Spark Cluster

This way we can easily and quickly read and manipulate the data from HDFS from a comfortable environment such as Jupyter. The following examples show how we can effortlessly query gigabytes of data:

```
In [90]: %%spark
df = sqlContext.read.parquet("/user/picae/ccma/consolidat/")
df.where(F.col("live")=='diferit').groupby('usuari_id').count().agg({"count": "avg"}).show()
```

```
+----------------+
|      avg(count)|
+----------------+
|6.75287203884977|
+----------------+
```

Figure 7.2: Average interactions per user_id

```
In [202]: %%spark
df.where(F.col("ui").isNotNull() & (F.col("live")=='diferit')).groupby('ui').count().agg({"count": "avg"}).show()
```

```
+------------------+
|        avg(count)|
+------------------+
|24.385671175101802|
+------------------+
```

Figure 7.3: Average interactions per logged user

7.2 ALS

The first solution that we explored was using the implementation of a collaborative filtering solution by the Spark machine learning library `MLlib`. Spark has a whole

library of machine learning algorithms that cover virtually all the needs in the landscape. In our case we will make use of the `ALS` module, which provides a distributed algorithm to calculate the latent factors that will be used to approximate the rating matrix.

This algorithm relies on matrix factorization to perform the task and, as expected by the name, uses *Alternating Least Squares* to learn the latent factors. It only requires an input dataframe and the name that correspond to the columns that contain the information about the `user_id`, the `item_id` and `rating`.

In our particular case, users do not have the capacity to rate the videos that they watch on the platform. For to this reason we will treat all interactions as positive interactions, since the user made an active effort to find and watch the content they chose. Therefore, all ratings are 1. The fact that all interactions are treated as positives brings us to the following question: do we need *negative* interactions in order to have a balanced set?

The question makes a lot of sense because interactions can tell us about what the users likes —positive interactions— but also about what the user does not like —negative interactions. This is why, in recommender datasets it is always advised to have a balanced set of positive and negative interactions for each user.

If the dataset that we have at hand does not have negative interactions —like ours— we might need to fabricate our own. The best way to do this is by selecting items that are very popular but the that user has not interacted with. It is generally believed that the fact that an item is very popular but the user has no interaction with it means that the user is not interested in the item. As for the unpopular items, the user may not find the item on the website at all, so it's not interesting either way.

Hence, going back to the question, the answer is no, and it due to the way `ALS` treats unseen entries: it is meaningless to manually add negative samples — samples with $R_{ui} = 0$— because the missing value / non-observed value is 0 for the algorithm as well, indicating that the user did not interact with the item. We can see a practical example in [28].

After clearing that up we can go ahead and create the recommender.

```
1 (training, test) = df.randomSplit([0.8, 0.2])
2 als = ALS(maxIter=5, regParam=0.01, userCol="indexUser", itemCol="
      indexItem", ratingCol="rating")
3
4 model = als.fit(training)
5
6 predictions = model.transform(test)
7 user_recs = model.recommendForAllUsers(10)
8 user_recs.show()
9
```

```
evaluator = RegressionEvaluator(metricName="rmse", labelCol="
    rating",
                                predictionCol="prediction")
predictions = predictions.where(f"prediction != \'NaN\'")
rmse = evaluator.evaluate(predictions)
print(rmse)
```

Listing 7.1: Spark ALS

As is apparent in the code example, the input columns for both items and users are called `indexUser` and `indexItem`. This is because previous to being fed to the recommender it is necessary that the input columns be in integer type because `ALS` only supports integer identifiers. Therefore, we link each `user_id` to an integer index and we do the same for `item_ids`.

Also, the values of the parameters have been set to the recommended values, but the correct course of action would be to find the best values for them via the use of hyperparameter tuning —implemented in spark with `CrossValidator` and `TrainValidationSplit`.

The reason why we take only those predictions that are not empty is because due to the random nature of the split of the training and test set, it is actually very common to encounter users and/or items in the test set that are not in the training set, making it impossible to predict a rating.

Finally, this model trained with the dataset resulting from all the analysis from 5 yields a $RMSE = 0.1891$.

In the following image 7.4 we can see a list for 10 users and the top recommendations for each user.

```
+---------+--------------------+
|indexUser|     recommendations|
+---------+--------------------+
|      148|[[35169,2.0652566...|
|      463|[[35169,2.759345]...|
|      471|[[35169,2.7320018...|
|      496|[[47313,1.6315632...|
|      833|[[35169,2.5081866...|
|     1088|[[35169,2.876184]...|
|     1238|[[35169,2.238747]...|
|     1342|[[35169,2.6739829...|
|     1580|[[35169,2.5030658...|
|     1591|[[35169,2.6282873...|
|     1645|[[35169,2.1219182...|
|     1829|[[35169,2.5031412...|
|     1959|[[35169,2.8805993...|
|     2122|[[35169,2.6274588...|
|     2142|[[35169,2.5226188...|
|     2366|[[47525,2.1030085...|
|     2659|[[35169,2.8716972...|
|     2866|[[35169,2.80185],...|
|     3175|[[36923,1.299705]...|
|     3749|[[35169,2.9269936...|
+---------+--------------------+
```

Figure 7.4: Recommendations for 10 users

7.3 Light FM

As explained in previous sections, the idea of the whole project is to create a recommender system that can take advantage of information derived from two other processes: user profiling and content indexation. Both of these will output feature vectors that can be associated with content —video in our case— and users —the viewers.

As we saw in the previous section 7.2, `ALS` takes users and items as inputs as well as the rating of their interaction. However there is no possibility to take advantage of other information that we might have available. This is why the `ALS` recommender is only half useful: it performs adequately but doesn't take advantage of the whole data that could take recommendations to a superior level.

This is why a recommender system based on Factorization Machines was necessary: the need to integrate all of the resources that we have at our disposal. At the current state of the project we do not have yet implemented the pipelines nor the complete processes to extract the features from users and contents. In order to make use of these assets once they are computed in the future they are to be stored in an Elasticsearch index.

The chosen library, for a first iteration at least, was `LightFM`. Since the dataset that we use is the same as the previous solution the same concern about the negative samples arises.

Contrary to the previous algorithm, in `LightFM` the embeddings are learnt through stochastic gradient descent methods. This method, as explained in 4.2.2, is a generic method to optimize an objective function. `LightFM` offers 4 different objective functions —logistic, Bayesian Personalised Ranking (BPR), Weighted Approximate-Rank Pairwise (WARP) and k-OS WARP— that can be used depending on the nature of the data at hand.

In our case, and as the documentation says, we use a Bayesian Personalised Ranking objective function because it is specially useful when only positive interactions are present. BPR maximises the prediction difference between a positive example and a randomly chosen negative example, understanding by negative example an interaction that has not been observed in the training set. Since this algorithm produces its own negative samples through sampling we do not need to generate negative samples either.

That being said, let us quickly explain the necessary steps to create a model using `LightFM`. The first step is, similar to the indices from section 7.2, to create a mapping between the user and item ids from our input data to indices that will be used internally by our model. We do this because `LightFM` works with user and item ids that are consecutive non-negative integers. The `Dataset` class allow us to

create a mapping between the IDs we use in our systems and the consecutive indices preferred by the model. The method `fit` does that for us:

```
dataset = Dataset()
dataset.fit((row['user'] for _, row in train_set.iterrows()),
        (row['item'] for _, row in train_set.iterrows()))
```

Listing 7.2: Id mapping for items and users

The second step is to build the interaction matrix.

```
(interactions, weights) = dataset.build_interactions(
                            [(row['user'], row['item']) for _, row
                                in train_set.iterrows()])
```

Listing 7.3: Creating the interaction matrix

The idea is to create a list of tuples containing which user has interacted with which item. The same thing can be done in order to create the features for both users and items:

```
item_features = dataset.build_item_features(
                            [(row['item'], row['tematica']) for _,
                                row in train_set.iterrows()])
```

Listing 7.4: Creating the features for items

In this case we add, as an example, the thematic of the content, but we could add anything that is related to the content and that we might think will be useful.

Finally, we have all the pieces that we will need to build the model:

```
model = lightfm.LightFM(loss='bpr')
model.fit(interactions, item_features=item_features, epochs=30,
    num_threads=16)
```

Listing 7.5: Creating the features for items

As we can see we are using BPR as loss or objective function, leaving the remaining parameters to their default value. Just like with `ALS`, we could do some hyperparameter tuning to find the best combination of parameters. The `epoch` parameter tells us how many iterations at maximum it will do before stopping and `num_threads` tells us on how many threads the process will run —it is set at 16 because the CPU from the NexTReT server has 16 cores.

Now, to assess the performance of the recommendations we used Hit Rate. In the metrics chapter (3.1) we can see a detailed implementation of the method to calculate this metric.

Once we have trained the model we can apply the hit rate algorithm. First, as a test, we trained the model using the `user_id` as id for the users, instead of the id for logged on users `ui`. With this configuration we obtained the following results:

- **`HR@5`**: 3.58%
- **`HR@10`**: 7.86%
- **`HR@25`**: 14.38%

Let us see that using one identifier or another makes a difference. The following results are obtained using `ui` as the identifier for the users.

- **`HR@5`**: 4.32%
- **`HR@10`**: 8.98%
- **`HR@25`**: 15.90%

Although these results might not seem very good at first glance we have to consider that we are looking for a very specific item among a set of more than 15000 items. Also, as expected, we see and improvement of 20% just by using the better identifier.

In terms of time performance, the model trained in 16 minutes and 12 seconds and the metric was calculated in 8 minutes and 43 seconds. Hit rate is an expensive metric because for each sample in the test set —1 for each user— it requires to calculate the first 25 recommendations and check if it is in the first top 5, 10 and 25.

Chapter 8

Conclusions

After working in this project the biggest takeaway that I got was how little of the data we can actually take advantage of. After a very thorough analysis and exploration of the data to find out what we actually consider useful, we end up considering only 2.6 million rows out of the total 1700 million. This is due to the policy that the CCMA has about their content: allowing everyone access all of the content either if they are registered or not.

If the CMMA wants to improve the range of people that the recommender can reach they should really consider changes in their content availability policy. For example, making some content exclusively accessible for logged on users or only making it watchable for not registered users some time after it has been released — temporarily exclusive. In short, actions to encourage users to register to the platform to increase the base of registered users because at the present time there are very little incentives for a user to create an account.

Another important point that has become really clear to me is how difficult it is to asses the performance of a recommender system. While other predictive methods can use typical metrics such as RMSE, Precision, Recall, F1 score, R^2 ... in Recommender Systems we have to really be sure what we are looking for in our recommender. Most of the metrics are not really related. For instance, a really diverse recommender might not be really accurate and also offer really small coverage. It is often difficult to optimize many metrics at once, which makes it so that we have to decide what we want our recommender to excel at. And in our case, what metric is the most important to take into account?

Since the CCMA is a public entity, their main concern is not economic but to provide the best service to their users and also promote their own content. Considering this, a diverse recommender would achieve both. Serendipity is always a goal for a recommender but is really difficult to target.

8.1 Future work

At this moment the project is still in a really premature stage. Out of the three main technical problems (1.2.2) none of them are completed or close to being so. For this reason it is difficult to see the progress of the whole ensemble because all of the pieces are incomplete and, therefore, do not fit. So far our work has served to create the base for a next level recommender system. A study of the state of the art of recommender systems has been done and has been essential to find and understand the best implementation from the huge variety of recommenders that there is. It has also been key, combined with the conducted data exploration, to discard the previous ideas for a sequential recommender. Also, the study in the metrics that will be used to evaluate the recommender will be necessary due to the special implications of developing a recommender for a public entity.

The tasks that are pending are:

- **User Profiling**: This ongoing task plans to create a set of features for each user.
- **Content Indexing**: Similar to the previous task, it is expected to create a set of features for the videos that can be consumed by users.
- **Joint Recommendation**: using the previous two assets jointly with a factorization machine recommender system like the one described in section (7.3).

The terrain has been explored and prepared for this curated information to be fed into the recommender. However, some architectural work also needs to be done to store the features once they have been computed. These, as a first consideration, will be stored in an ElasticSearch index, one for user features and another for content features. The latter will most likely be stored in the already existing content index, jointly with the content metadata explained in section (5.2). Setting up all these structures and subsequent processes that will later query the required data when needed —probably when creating the interaction matrix to add the features— is also a long and key task for the project.

Other concepts to studied in the future are ideas the could hinder the performance of the recommender but that are not flaws or characteristic of the recommender itself:

- **Failing Importance**: Most recommenders do not consider whether the recommendation they made actually worked. This is why, in the future, the fact that the top recommendation for a user was seen and not selected should penalize that content in order to improve recommendation in the future for that user. This impact should be studied in the future.

- **Cannibalization**: The recommender section in the web page is not at the top. Hence, a lot of content links can be seen and accessed before scrolling down enough to see the recommender section itself. It could happen that a user sees a content that is shown in the page —such as the new episode of the tv series that they follow— and clicks it to watch it before having reached the recommender section. There is the possibility that the content that they clicked was on the recommendation list. Therefore, they watched a recommended content but not thanks to the recommender, although it probably would have worked if the recommender had shown up in a higher part of the web page.

References

[1] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor, "Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems," 09 2015. [Cited on page 6]

[2] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," pp. 7–10, 09 2016. [Cited on page 6]

[3] M. Ekstrand, D. Kluver, F. Harper, and J. Konstan, "Letting users choose recommender algorithms," pp. 11–18, 09 2015. [Cited on page 6]

[4] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," pp. 17–22, 09 2016. [Cited on page 7]

[5] Y. Sun, N. Yuan, X. Xie, K. McDonald, and R. Zhang, "Collaborative nowcasting for contextual recommendation," pp. 1407–1418, 04 2016. [Cited on page 7]

[6] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, "Context-aware recommender systems," *AI Magazine*, vol. 32, pp. 67–80, 09 2011. [Cited on page 7]

[7] M. Braunhofer, I. Fernández-Tobías, and F. Ricci, "Parsimonious and adaptive contextual information acquisition in recommender systems," 01 2015. [Cited on page 7]

[8] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," pp. 635–644, 01 2011. [Cited on page 7]

[9] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," pp. 301–304, 10 2011. [Cited on page 7]

[10] A. Odic, Ante, TKALČIČ, Marko, TASIČ, J. F, A. Kosir, and Andrej, "Predicting and detecting the relevant contextual information in a movie-recommender system," *Interacting with Computers*, vol. 25, pp. 1–17, 01 2013. [Cited on page 7]

[11] E. Çano, "Hybrid recommender systems: A systematic literature review," *Intelligent Data Analysis*, vol. 21, pp. 1487–1524, 11 2017. [Cited on pages 9, 10, 11, and 12]

[12] R. Burke, "Knowledge-based recommender systems," in *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, p. 181–201, Marcel Dekker, 2000. [Cited on page 10]

[13] A. Felfernig and R. Burke, "Constraint-based recommender systems: Technologies and research issues," *ACM International Conference Proceeding Series*, p. 3:1–3:10, 01 2008. [Cited on page 10]

[14] C. C. Aggarwal, *Recommender Systems - The Textbook*. Springer, 2016. [Cited on pages v, 11, 13, and 34]

[15] L. de Campos, J. Fernández-Luna, J. Huete, and M. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks," *Int. J. Approx. Reasoning*, vol. 51, pp. 785–799, 09 2010. [Cited on page 12]

[16] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, K. Fujimura, and T. Ishida, "Classical music for rock fans? novel recommendations for expanding user interests," pp. 949–958, 10 2010. [Cited on page 22]

[17] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 4511–5, 02 2010. [Cited on page 23]

[18] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender systems handbook*. New York; London: Springer, 2011. [Cited on pages 23 and 25]

[19] S. Vargas and P. Castells, "Rank and relevance in novelty and diversity metrics for recommender systems," pp. 109–116, 10 2011. [Cited on page 23]

[20] C.-N. Ziegler, Cai-Nicolas, S. McNee, S. M, Konstan, J. A, Lausen, and Georg, "Improving recommendation lists through topic diversification," 01 2005. [Cited on page 23]

[21] D. Kotkov, S. Wang, and J. Veijalainen, "A survey of serendipity in recommender systems," *Knowledge-Based Systems*, vol. 111, 08 2016. [Cited on page 24]

[22] T. Murakami, K. Mori, and R. Orihara, "Metrics for evaluating the serendipity of recommendation lists," vol. 4914, pp. 40–46, 12 2008. [Cited on page 24]

[23] M. Ge, C. Delgado, and D. Jannach, "Beyond accuracy: Evaluating recommender systems by coverage and serendipity," pp. 257–260, 01 2010. [Cited on pages 24 and 25]

[24] Y. Zhang, D. Séaghdha, D. Quercia, and T. Jambor, "Auralist: Introducing serendipity into music recommendation," pp. 13–22, 02 2012. [Cited on page 24]

[25] Google, "Matrix factorization; recommendation systems; google developers," Feb 2020. [Cited on page 37]

[26] D. Wagner, "A generalized birthday problem," in *In CRYPTO*, pp. 288–303, Springer-Verlag, 2002. [Cited on page 46]

[27] Catly_hit, "Explainable recommendation: A survey and new perspectives," Aug 2018. [Cited on pages v and 53]

[28] Vinta.ws, "Negative sampling; vinta.ws," May 2017. [Cited on page 57]