MASTER'S THESIS 2020

Smart Personalization for In-flight Entertainment Systems

Sara Trygve, Frida Gunnarsson

ISSN 1650-2884
LU-CS-EX: 2020-50

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-50

Smart Personalization for In-flight Entertainment Systems

Sara Trygve, Frida Gunnarsson

Smart Personalization for In-flight Entertainment Systems

Sara Trygve
dic14str

Frida Gunnarsson
tys14fgu

August 20, 2020

Master's thesis work carried out at Tactel AB.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Sören Just Pedersen, soren.justpedersen@tactel.se
Johan Liljeros, johan.liljeros@tactel.se

Examiner: Jörn Janneck, jorn.janneck@cs.lth.se

Abstract

In-flight entertainment systems are becoming more advanced and a shift towards a more personalized experience can be seen in the industry. A challenge with providing personal recommendations for new passengers is the lack of initial user information. This problem is often solved with a login or a connection to a users own mobile device. However, this solution only covers a small amount of customers.

In this thesis, we focused on creating a personalized experience for the rest of the passengers. We investigated different approaches to provide recommendations, such as content-based recommendations and association rule mining. We used techniques such as one-hot encoding, word embeddings, and Apriori.

Using data from past passenger interactions with existing IFEs, we evaluated the experiments by a comparison of a given recommendation and previous watched item. The final model gives category recommendations as well as ranks the items after similarity compared to previous watched content. When filtering out the empty recommendations, we could reach an accuracy of 30 %.

Keywords: In-flight entertainment, recommendation systems, personalization, machine learning techniques

Acknowledgements

We would like to thank our supervisor Pierre Nugues for his invaluable guidance, technical support and ideas during this project. We would also like to thank our supervisors at Tactel, Sören Just Pedersen and Johan Liljeros, for very helpful feedback and insights.

To all employees at Tactel, thank you for a warm welcome and for giving us the opportunity to carry out this interesting project in collaboration with you.

Finally, we would like to thank Stina, Ebba and Hanna for endless positivity, laughter and many enjoyable “fika” breaks during our theses.

Contents

Chapter 1
Introduction

During long-haul flights, a screen on the back of the seat in front of each passenger is not an uncommon sight. These are called *in-flight entertainment* (IFE) systems and contain different entertainment options, such as duty-free shopping, maps over the aircraft's route, and various media content, such as movies, TV shows, music, and games. In-flight entertainment systems are not a new concept, but are becoming more complex, with larger content range and more varied activities for the passengers to interact with. Personalization is widely used today by e-commerce websites, media platforms, and online advertisements. It is a way to propose a customer recommendations relevant to her/his previous actions or previously bought items. An individual customer can be identified using cookies, login or connection to other accounts. It provides a more personal experience and possibly more sales.

Creating a mobile application, which can be connected to the IFE is one approach to personalize the experience of in-flight entertainment systems. With such an application, the passenger can create his or her own profile and save movies to watch during their next flight. However, this requires the creation of an account, as well as planning and downloading the application in advance since this can't be done during a flight due to no or limited connection. While this is a great solution for the passengers who take this additional step, we will focus on the remaining passengers. In this thesis, we will explore different ways to provide a solution for personalization during the flight without a login and without much data on each passenger. In addition, we will focus on the large variation of available media content.

During this Master's thesis, we created a system that targets the users who did not download the application *i.e.* a system that could give personalized suggestions without the need of creating a login. Trending topics such as big data makes this possible. We approached this problem by looking at patterns of the collected in-flight data to be able to make recommendations according to what other passengers have previously watched during their flights.

Research Questions. In this thesis, we will explore and discuss the following questions:

- How can an IFE be personalized during flight with limited information?
- How effective are the different approaches to personalize an IFE and can they be combined?
- What additional feedback system can be used to make the recommendations more accurate?

Approach. To answer these question, our work consists of three main steps:

1. First, we researched the existing approaches of personalization and recommendation systems within in-flight entertainment.
2. Then, we implemented a recommendation engine to recommend content to each individual passenger. To do this, we studied two main techniques:
   - A content-based movie recommendation system, which calculates similarities between movies, and
   - A market basket analysis, derived from a dataset provided by Tactel.

   We computed the accuracy and usability of these models.
3. The last step was to combine the two different approaches into a working recommendation system.

We carried out this thesis at Tactel AB, a digital interaction agency, at their head office in Malmö. Since 2015, Tactel is a part of Panasonic Avionics Corporation. They have been working with different in-flight entertainment systems for several years and could provide the knowledge and data necessary to carry out this project.

Contribution. We carried out this thesis in full collaboration. Both of us have participated in the research, implementation and documentation of all parts of this project. During the implementation of this project, a lot of discussions and pair programming were done, for both of us to have a better understanding of the final result.

However, the main responsibility for some parts was divided between us. For example, when working in parallel with content-based recommendation and association rule mining, Sara was in charge of content-based recommendations, and Frida of association rule mining. During the combining of the two models, Sara implemented the function to add music to our model, while Frida added TV shows.

During the later parts of the project, Sara implemented the test interface with movies, while Frida made the design of the interface suggestions.

The writing of the report was also divided similar to the implementation. Sara had the main focus on content-based recommendation and Frida on the association rule mining.

Chapter 2
Background

2.1 In-flight entertainment

The existence of in-flight entertainment system reaches back almost 100 years, when a promotional video was shown at a flight in 1921. Up until the eighties, the movies were shown on projectors and bigger screens shared by multiple passengers, while the sound was played from shared speakers throughout the cabin. But when LCD screens appeared on the electronics market, the in-flight entertainment evolved to seat-back mounted screens.

When further technical advancements were made, a distributed system which showed the same content to all passengers could be replaced by a digital video-on-demand (VOD) system. This allowed the passenger to control the movie with play, pause etc. Today, complex software solutions have been developed, and the functionality of the IFEs has been widely extended. Information such as the position, height, and speed of the flight can be viewed, as well as numerous movies, TV shows, games, duty-free shopping and broadcasted live news entertains the passengers during flights (White, 2012).

2.2 Personalization

Over the past few years, personalization has been a big trend among companies. According to a survey done by Evergate and Researchscape International, 98% of companies agree that personalization helps advance customer relationships. Even if a majority of personalization is done within emails and with more simple approaches, it can also be seen that the usage of machine learning in this area has gone up from 26% in 2018 to 40% in 2019. Most importantly, the personalizations show results — a lift of more than 10% is reported by more than half of the participating marketers. Another relevant note is that the companies using machine learning for personalization are more satisfied with the results compared to other approaches (Researchscape International, 2019).

2.3 Recommendation systems

Recommendation techniques are an important component of systems which handles a very large and wide content database. One of the main goals of a recommendation system is to increase sales by bringing relevant products to the attention of the user. Even though this is often the first reason any company would implement such systems, the experience of the user will be less cluttered with more accurate recommendations. S/he will avoid having to scroll through thousands of products and the overall impression will certainly be more satisfying.

Except relevance, there might also be advantages in recommending products unknown to the user (called *novelty*) or products seemingly unrelated i.e. products that the user still might appreciate (called *serendipity)*. Recommendation systems are often evaluated through ratings and the creation of training and test sets to compare different models (Aggarwal, 2016).

In the sections below, a few different techniques to implement recommendation systems are described.

2.3.1 Collaborative filtering

Collaborative filtering models for recommendation systems use ratings across multiple users. This results in identification of users with similar taste, and recommendation of new items according to previous ratings of other identified similar users (Aggarwal, 2016). See Figure 2.1.

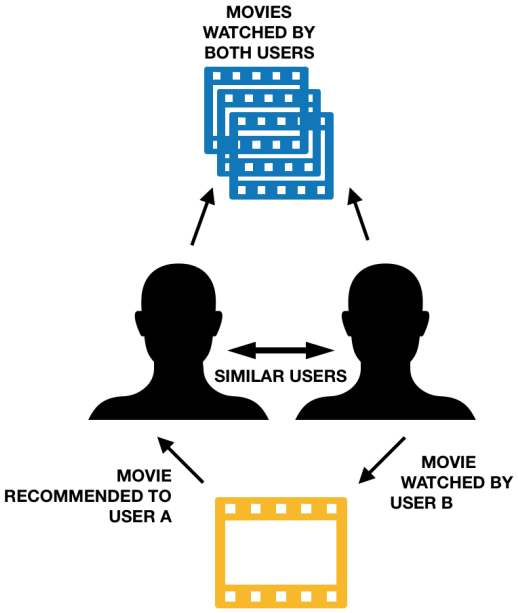


Figure 2.1: Collaborative filtering

2.3.2 Content-based filtering

Another approach for recommendation systems is to use content-based filtering, where instead the items attributes are used to make recommendations. The result of these recommen-

dations can be seen as more straight-forward since it is based on the keywords or content of the item. The idea is to not recommend an item with keywords which are not present in the items previously purchased by a customer. This might lead to reduced diversity of the recommended items. This approach is effective if there is a lot of data about the user's previous behavior. The more data there is, more item attributes can be used to find new matching items (Aggarwal, 2016). See Figure 2.2. However, if the user is new, with no previous consummations there is no data to make recommendations based on. This is called a *cold-start problem*, see Section 2.3.4.

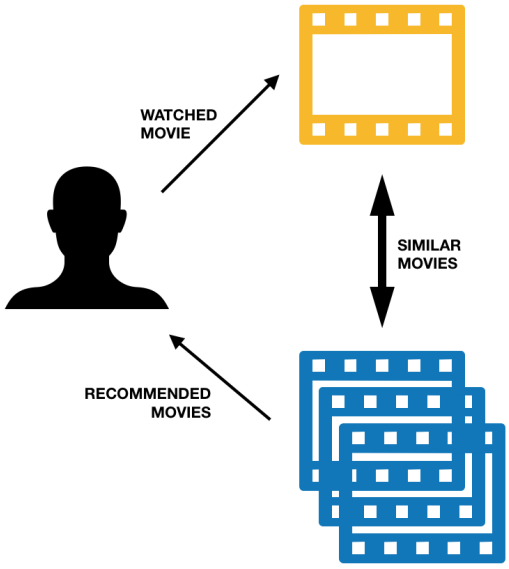


Figure 2.2: Content-based filtering

2.3.3 Hybrid recommendation systems

Most commonly, the two techniques mentioned above are combined to create a hybrid system that takes both the item similarity and the user similarity data into account. This provides the flexibility of using different types of recommendation systems for the same task (Aggarwal, 2016).

2.3.4 Cold-start problem

In an IFE without a login, every user will be a new user. There is no historical data about the current user and this becomes a classic cold-start problem. Shi et al. (2017) proposed two main approaches to deal with the cold-start problem in recommendation systems – content-based recommendations and interview-based:

- For content-based, the metadata of the items are used to recommend similar items to the user.
- Interview-based uses a few initial questions to profile the user and get a hint of their preferences (Shi et al., 2017).

2.3.5 Feedback

There are two main types of feedback data; implicit and explicit. With explicit feedback a rating system is used, and each customer can rate the product depending on the experience they had. This gives more information compared to implicit feedback, which does not use any rating scale, instead it is implied that if the user watched the movie or bought the item, it is considered to be positive feedback (Jawaheer et al., 2010).

For recommendation systems, explicit feedback from the users is an important factor, since the watched content or bought item can be both liked and disliked. It is preferred to give recommendations similar to the products the customer gave a higher rating and avoid recommending products the user gave a very low rating. Many recommendation systems also use the ratings to calculate the accuracy of different recommendation engines, where the goal of the model is to predict an item's rating. It can then be compared to the actual rating of the item (Piotte and Chabbert, 2009).

2.4 Related work

In this section some relevant resources which we have received inspiration and knowledge from will be briefly discussed.

2.4.1 Movie Recommendation

Netflix Grand Prize. In 2009, there was an important step of improvement of movie recommendation systems when Netflix held a competition to improve their existing recommendation engine *Cinematch*. The goal was to predict the users rating of a movie, and the winning team BellKor's Pragmatic Chaos managed to do so with an accuracy of over 10% increase from Cinematch.

A joint solution of three teams (BellKor, Pragmatic Theory and Big Chaos) gave a quiz set RMSE (root mean square error – the standard deviation of errors) of 0.8554. This solution contains multiple different techniques, for example clustering, classification, Restricted Boltzmann Machines, KNN, regression and more. However, as all the teams clarify, their solution is probably not practical for a real-life recommendation system because of its complexity and the fact that performance was not a factor when developing the most accurate rating predictions (Piotte and Chabbert, 2009; Töscher and Jahrer, 2009; Koren, 2009).

Plot-based Movie Recommendation. Tang et al. (2020) who created the Weighted-PathSim model explored the possibilities of recommending a movie based on plot similarity. Some of the most common features used for content-based movie recommendation are genres, actors, and directors. However, Tang et al. (2020) argued that genre is too general. To solve this issue, Tang et al. (2020) explore three ways of handling plot description – phrase mining, topic modelling and similarity measure:

1. Phrase mining is done to find keywords or phrases from a plot that can be compared to other movies;

2. To eliminate that the keywords have to be exactly the same to get a match, topic modelling is done by clustering similar words;

3. In the last step, the topic labels from step 2 are used to do a similarity search, by weighing a label by the number of times it appears in the description and then comparing movies either by using weighted or unweighted cosine similarity.

Actor embeddings. Another approach to improve content-based movie recommendations was discussed by Kim et al. (2019) who proposed a solution to group similar actors together. They create an actor model base on movies, genres, character roles, and descriptive keywords to make cast predictions and a versatility ranking of actors. This actor model could also be used in a content-based movie recommendation system, where similar actors could be associated together.

2.4.2 IFE Personalization

A flight can be a stressful situation for some people. Liu (2007) described this phenomenon in a report about research directions for in-flight entertainment systems and how a personalized system can help reduce the negative feelings of stress. Three main activities are solutions proposed to deal with stress during a flight:

1. Communication with other passengers,

2. Different types of exercises and

3. Relaxing with books, movies and other available material.

Most of these activities are available during long-haul flights, but the focus often lies within the last activity – different types of entertainment. One observation made by Liu is that these systems are often homogeneous and hard to navigate if containing many available options. On the other hand, if the content is very limited, the passenger cannot find anything desirable. In both cases, the IFE does not help to limit the stress. Liu proposes a solution which utilizes user profiles to create recommendations to better reduce the stress of the passengers. He also discusses the challenges with such a system. A user's Behavior and desires change over time, and to create a model that tracks and updates according to such changes comes with many difficulties.

According to Garcia (2016a), Lufthansa stated that the next step for in-flight entertainment is personalization and recommendations similar to the ones on Netflix. The motivation for this is to keep their customers more engaged. The ability to connect with a personal device and extended connectivity such as Wi-Fi are also future possibilities to increase the personalization. A connected device, or a companion application, helps bringing a passenger's viewing history from one flight to another, suggesting her/him to keep watching an unfinished movie. The IFE product manager at Lufthansa also comments on future possibilities to dynamically load content based on passenger data (Garcia, 2016a). There are ongoing discussions on data collection, mainly used to predict content to keep or discard for next month. Nonetheless, the same type of data could also be used for machine learning and recommendations (Garcia, 2016b).

In her thesis, *Dynamic airline in-flight entertainment systems using predictive analysis*, Hawk (2018) explores a similar idea, with a strong focus on monetization and value for the companies. She emphasizes the value of having an innovative IFE and explores ways of adapting them to improve each customers experience. Hawk describes how predictive analysis can be used to improve the content distribution network, by offering more popular content to help build customer satisfaction but also lower the costs for the airlines. Predictive analysis has previously been used to forecast demand for flights, estimate fuel consumption, predict travel preferences and crew activity among other things, and could be extended to IFE systems as well.

To include the IFE content in the predictive analysis, Hawk proposed three steps:

1. Defining passenger demographics,
2. Integrating airline application with third-party applications and
3. Identifying groups and preferences.

With these steps, personalized recommendations can be done according to the user profile. If a passenger does not complete all of these steps, Hawk's model will give recommendations according to destination. The conclusion of the implementation of predictive analysis is that the content of the IFE directly influences the passengers experience. This means that companies can save money by offering the right kind of content (Hawk, 2018).

Chapter 3
Approach

3.1 Methodology

In this thesis, we used the *Cross Industry Standard Process for Data Mining* (CRISP-DM) model as a method to conduct the project. CRISP-DM is a structured approach that defines a process model, which provides a framework for carrying out data mining projects in a structured way. This method aims to make the project less costly, more reliable, more manageable, and more repeatable. The model provides an overview of the life cycle of a data mining project.

CRISP-DM describes six different phases of the process and their tasks. The different phases can be performed in a different order and it's often necessary to backtrack to previous phase and repeat certain actions. This is shown by the arrows in Figure 3.1. The phases are describes in more detail below (Wirth and Hipp, 2000).

Business understanding. The first phase is focused on understanding the business and project requirements, in order to design a preliminary project plan. The knowledge can then be used to define a data mining problem. In this project, business understanding consists of discussions with the company as well as research of related work, which can be seen in Section 2.4.

Data understanding. Since the project plan and the data mining problem formulated in the previous phase also needs some kind of understanding of the data, this phase and business understanding are closely correlated with each other. This phase consists of collecting and getting familiar with an initial dataset. This includes identifying potential data quality problems and detect subsets that could be of interest. This is done in Section 4.3.1.

Data Preparation. To be able to use the data for modeling, the data needs to be prepared into the final dataset. This is likely to be performed multiple times and could include attribute selection, construction of new attributes, transformation of data for the

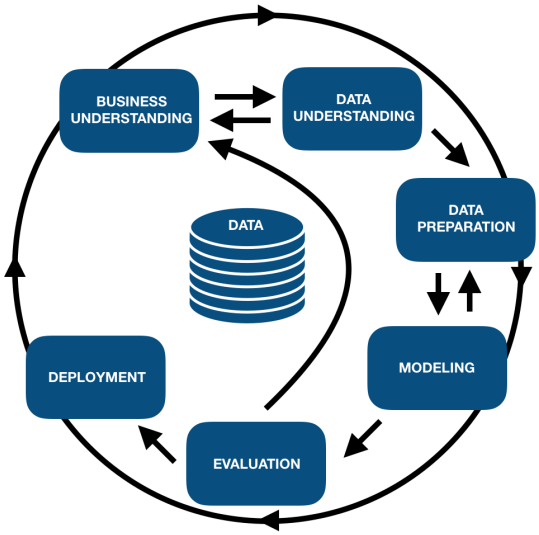


Figure 3.1: CRISP-DM model

modeling tools and data cleaning. We describe the necessary preparation steps in Section 4.3.1.

Modeling. In this phase, a set of modeling techniques are selected. Usually this can be done in combination with data preparation, since some problems with the data can be found while modeling. Chapter 5 describes the algorithms and models used throughout our project.

Evaluation. After building one or more models, it is important to thoroughly evaluate the models and review the steps executed to construct the models. The evaluation methods are described in detail in Chapter 6.

Deployment. The knowledge gained from the models will need to be organized and presented to the end user. This could for example be done by creating a report or implementing a system. This report is our result of this step, combined with an implementation of the model, but it has not been integrated with an existing system.

3.2 Utilities

The tools that we used during the implementation of the models will briefly be described below.

Hardware. MacBook Pro, 2.8 GHz Intel Core i7 and 16 GB RAM.

Python libraries. We chose python as the main programming language used throughout the implementation process together with various helpful libraries and modules. Below we describe the most important ones and how they were used:

pandas – We used the pandas library to sort the data in a DataFrame, which is an easy way to effectively handle multiple rows of data (Wes McKinney, 2010).

scikit-learn – We used this Python module for various machine learning tasks throughout the project, for example to one-hot encode the data and to calculate the cosine similarity between objects (Pedregosa et al., 2011).

mlxtend – An open source python library for machine learning and data mining tasks. During this project it was used for frequent pattern mining, the Apriori algorithm (Raschka, 2018).

Amazon Web Services. The provided in-flight entertainment data was collected through SQL queries in Amazon Web Services (AWS). To be able to access this data and run the models, we used AWS SageMaker[1].

[1]Available at `https://docs.aws.amazon.com/sagemaker/`

Chapter 4

Datasets

In this chapter, we describe the different datasets used in this project. These were both data provided by Tactel as well as public datasets for movies, TV shows and music.

To develop an effective recommendation system, it is important to take into account what kind of data and user feedback is available. None of the datasets used in this project contain any ratings from users. This means that they only contain implicit instead of explicit feedback, as explained in Section 2.3.5.

4.1 The Movie Database (TMDb)

For the movie recommendation models, we used the existing TMDb dataset[1] with 5,000 movies, consisting of all relevant information needed, such as title, cast, director, genres and a short overview for each movie.

The dataset is divided into two different subsets of the data, movies and credits, and therefore the data was preprocessed before it was used to build the models. We combined the two subsets into one to create a DataFrame consisting of the following columns: *Title*, *Actors*, *Director*, *Genres*, and *Overview*. Figure 4.1 shows the distribution by category.

4.1.1 TMDb API

After creating the first models, the movies which were available on the flights did not exist in the previously used limited dataset from TMDb. Therefore, we used a Python library called *tmdbsimple* to access the TMDb API[2] and thereby fetch all movie data necessary for our combined model[3]. This enabled us to access the complete data of TMDb, including the

[1] Available at: `https://www.themoviedb.org/`
[2] Available at `https://developers.themoviedb.org/`
[3] Available at `https://pypi.org/project/tmdbsimple/`

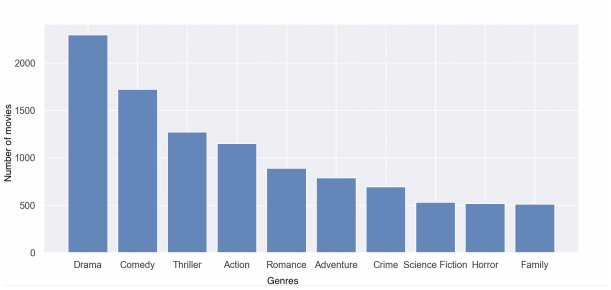


Figure 4.1: Histogram of genres in TMDb dataset

features used in the models mentioned above. This data also included similar information for TV shows, which made it possible to add this category to the recommendations as well.

4.2 Spotify API

To find the available music content on each flight, we sorted out all music that passengers had listened to during that flight. We then used the Spotify Web API[4] to fetch data for each item in that list. Using this dataset, we could create models similar to those used for the movies, and give recommendations for music as well. To fetch the data, we used a Python library for the Spotify Web API, called Spotipy[5].

4.3 In-flight entertainment data

The dataset provided by Tactel contains information about each specific flight, as well as media data for the passengers interactions with the IFE. Each flight is labeled with a flight ID which is specific to this dataset and can therefore not be connected in any way to the airlines data. Because of this, the individual user cannot be identified or connected to a real person in any way. All individual users are sorted out using a flight ID and a seat number. Information about the watched movies, podcasts, music and games per passenger can be listed and associated to one session, i.e. all content interacted with by one passenger during one flight.

The content is divided into three levels of categories; high level (e.g. 'Entertainment'), mid-level (e.g. 'Movies') and low-level ('Latest'). The media type is also stored in another column, divided into video, audio, or game. This was used to sort the data by the specific content types.

This dataset was also used to collect the available content on each flight with help from the flight ID. This enables the models to give recommendations for categories or movies that actually exist on the passengers flight. The complete selection of content for each flight was not available during this thesis. As a solution, all the media interacted with at least once during each flight was collected to create the set of existing media per flight. This means that some content might be missed, if no passenger had interacted with it.

[4]Available at `https://developer.spotify.com/documentation/web-api/`
[5]Available at `https://spotipy.readthedocs.io/en/2.12.0/`

The total size of the data is roughly 2.5 TB, or 25 billion data entries. We did the experiments with only one million entries, approximately 200 MB or 0.01% of the total data, due to limited processing ability of our computers. To increase the ability and accuracy, we explored different ways to run the created models in AWS SageMaker. However, due to limited RAM on the available instance at AWS SageMaker, this did not increase the computational ability of the model, and due to this the experiments were limited.

4.3.1 Exploratory Data Analysis

We carried out an exploratory data analysis to get an idea of the structure of the data, and how it was cleaned before the future experiments. In this section, we present some statistics and graphs to explain the spread between different levels of categories.

An initial check with 100,000 data points showed that only 20,366 (20%) out of these contained any media content information and, out of this share, there was 5,818 individual passengers who watched any type of media content. Since the only data of interest was where any type of content had been watched, we filtered the data in the initial SQL query to avoid this part of unusable data where `media_contenttype` was empty.

The resulting dataset of one million (1,000,000) data points is the data that we used for the subsequent experiments. The very low percentage that we actually used of the total data (0.004%) was due to limited processing power. With one million data points, we initially extracted 106,074 individual passengers. When removing the items with an empty `category_title`, which was the field most interesting to our experiments, this resulted in a total of 812,400 items and 97,299 individual passengers.

The mean value of number of watched items per passenger was at this point 8.35, but it was acknowledged that many of these were duplicates, such as the same movie three times in a row, probably due to play/pause actions by the passengers. We filtered out the duplicates and we obtained a mean value of 4.66. Removal of different items such as 'Welcome Screen', 'AD' and similar was also required. This lowered the number of individual passengers who actively used the screen to watch media to 86,364. The mean number of watched items per passenger was slightly decreased to 4.37.

The following lines show one example of the structure for a single data point:

```
["Crazy Rich Asians", "video", "Entertainment",
"Movies", "Latest", "Business",
"2018-12-08T06:18:47", "English", "20181208"]
```

Each item contains information such as date, movie title, category, and parent categories, see further description below. One session, identified by flight ID and seat number, contains a list of items like this. The dataset contains more columns, but these are most valuable during this project:

`media_title` – title of the media. Could be either movie title, artist name, song name or TV episode title depending on the media system used for the current flight;

`media_contenttype` – the content type of the media, could be either audio, video or game;

`category_parent_parent_title` – highest level of category, typically 'Entertainment';

`category_parent_title` – mid level category, for example 'Movies';

`category_title` – lowest level category, for example 'Latest', 'Kids', 'New Releases', 'TV';

`starttime` – the time when media content was started, could be used to order all watched content by each passenger;

`language` – media language or subtitles where applicable;

`date` – the date of the flight.

We calculated the graphs below with the dataset containing ten million (10,000,000) data points, which is 0.04% of the total data, to get a broader perspective compared to above mentioned statistics. Figures 4.4, 4.5, and 4.6 show the most popular categories for the three levels of categories. The different categories can vary between flights and their media system, and similar categories can be seen but with different names in the dataset, for example 'New Releases' and 'Latest'.

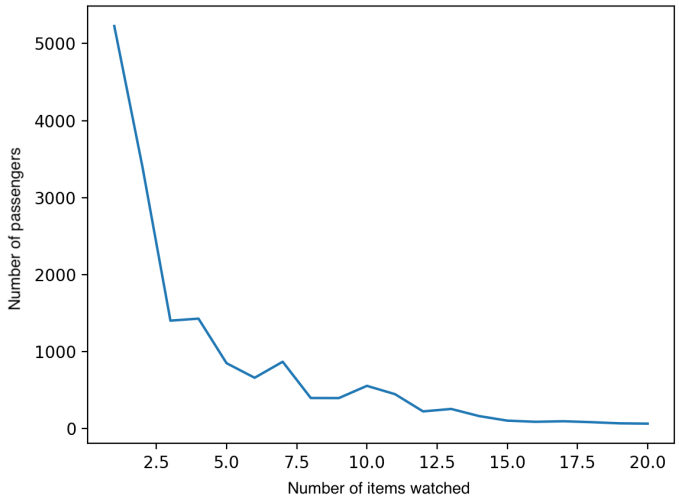


Figure 4.2: Number of watched content items per passenger

Data preparation

After looking at the statistics of the complete data as described in Section 4.3.1, we removed every data entry without any media content to create more efficient model computations as these were not usable for predicting recommendations. To be able to give useful recommendations, we also removed some content such as advertisements and welcome screens before training the models, as these were not considered helpful recommendations. A movie might appear several times for the same user, but we removed duplicates like this to not affect future models.

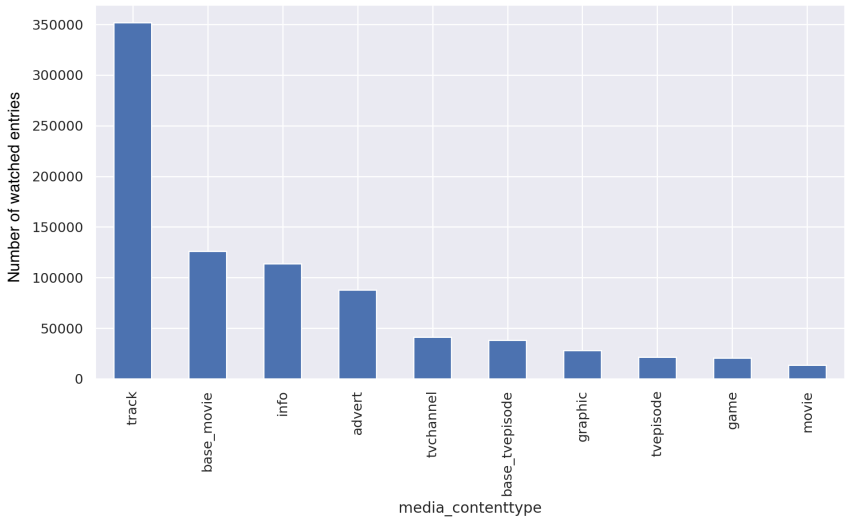


Figure 4.3: Histogram of media content

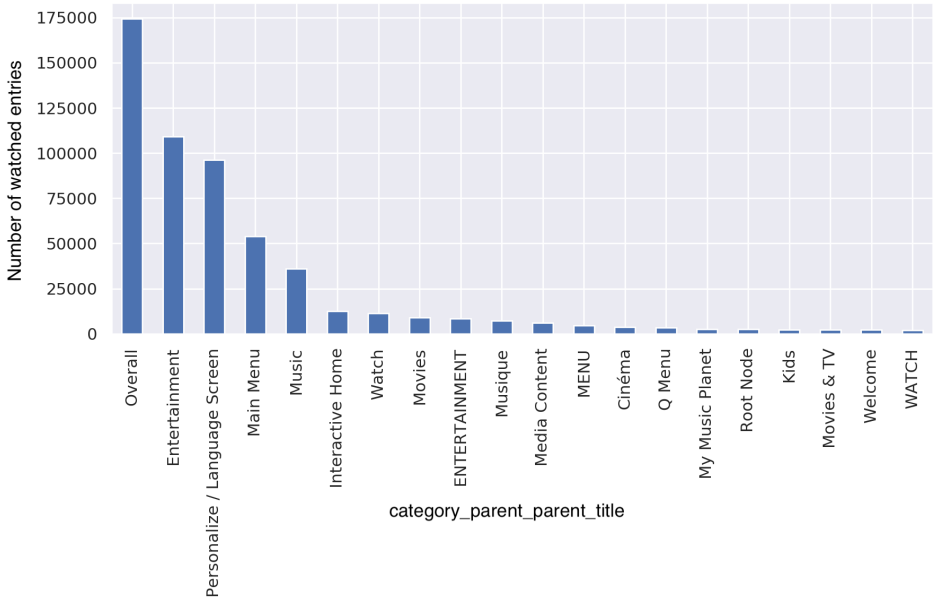


Figure 4.4: Histogram of top level category

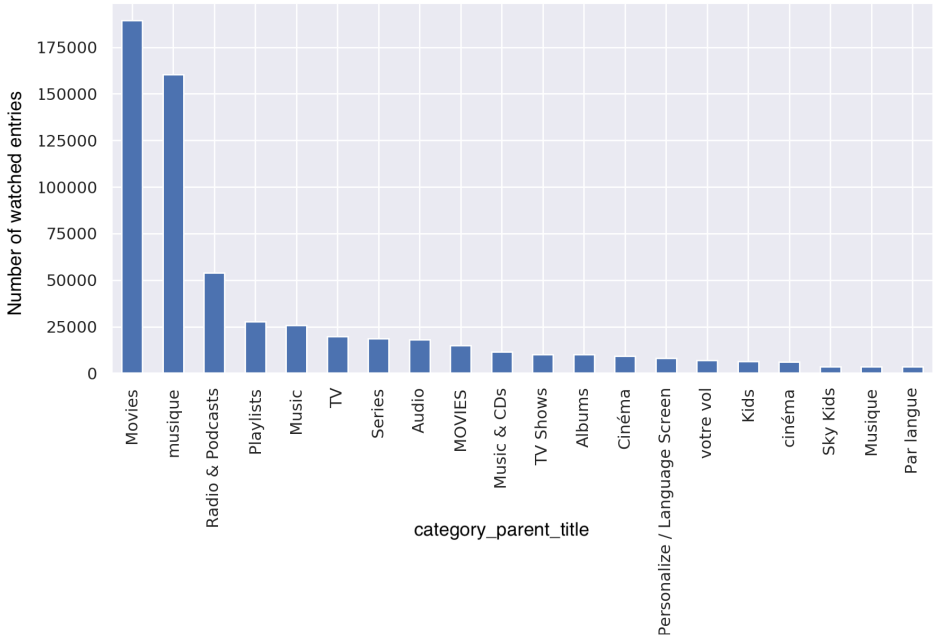


Figure 4.5: Histogram of mid-level category

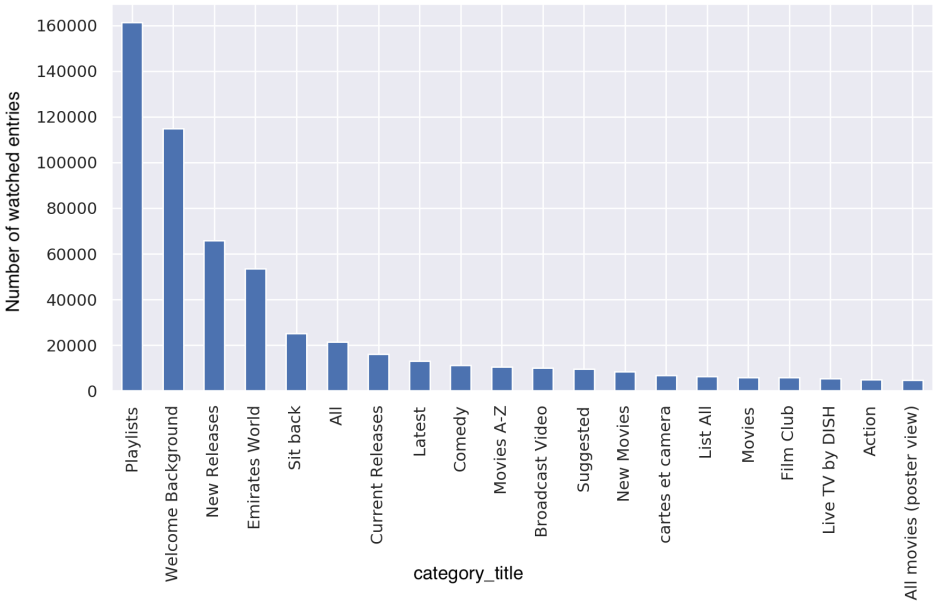


Figure 4.6: Histogram of lowest level category

Chapter 5
Algorithms and Models

In this chapter, we describe the algorithms used and the models which we created. This is divided into three main parts:

1. Content-based recommendations describes the calculations of movie similarity,
2. Market basket analysis uses data mining to create rules for correlations between items, and lastly,
3. These two models are then combined to create two levels of recommendations.

5.1 Content-based recommendation

The first step was to implement a recommendation system which focused on movies. We motivate this choice in reference to Hawk (2018) who mentions that this is the most popular activity as on-board entertainment for long-haul flights according to the IATA Global Passenger Survey from 2016. Later on, we adapted the same model for music and TV shows as well, to be able to give recommendations that covers more areas.

5.1.1 Metrics for similarity

We assessed two different metrics for distance measurement, cosine similarity and Euclidean distance.

Cosine similarity measures the angle between two vectors, and is calculated as the dot product of two normalized vectors, as shown in Equation 5.1. (Li and Han, 2013)

$$Similarity(A, B) = \frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.1}$$

Euclidean distance on the other hand, measures the ordinary distance between two points, and can be compared to measuring with a ruler in two- or three-dimensional space (Huang, 2008), shown in Equation 5.2.

$$Distance(A, B) = \sqrt{\sum_{i=1}^{n}(B_i - A_i)^2} \tag{5.2}$$

Some argue that using cosine similarity as distance measurement is more beneficial than Euclidean distance when the length of the items is not of importance. The reason for this is because cosine similarity measures the angle between two vectors and not the distance like Euclidean distance does. This is applicable to our models since it doesn't matter if the vectors have the same amount of, for example, actors. It is more important if the different vectors have one or more actors that are the same.

We assessed both metrics when creating the models in order to prove the theory that cosine was the right choice. To be able to do a quick comparison, we used an approach inspired by Tang et al. (2020) who created the Weighted-PathSim, to see how well the model gave recommendations of a movie that has sequels. For this test, we used the movie *Harry Potter and the Philosopher's Stone* as the initial movie. For the first attempts, the model using cosine similarity recommended five other Harry Potter movies amongst the top 10 movies. In contrary, the model using Euclidean distance recommended none of the other Harry Potter movies. This experiment strengthened the decision to use cosine similarity for future models. To evaluate our approaches further and see which model performed best, we carried out evaluation tests on 10 test users.

5.1.2 One-hot encoding

We first encoded the public datasets' different rows using one-hot encoding. In this way, the system could recommend the movies using similarities such as same director, actors, genres and words in a plot description. One-hot encoding means translating the values in the different rows into a binary vector, where a 1 stands for match and a 0 for no match.

An example of how a one-hot encoded DataFrame for genres look like is shown in Figure 5.1. In this figure, we can see that movie with index zero, has the genre *Action* and *Advednture*, but not *Animation* or *Comedy* etc. By comparing the different movie vectors, we can detect and calculate similarities. The cosine similarity was calculated separately for each one-hot encoded vector (genres, director, actors). Next, we calculted the mean value of all the cosine similarities, which made each vector affect the model equally (Zhang and LeCun, 2017).

	Action	Adventure	Animation	Comedy	...	TV Movie	Thriller	War	Western
0	1	1	0	0	...	0	0	0	0
1	1	1	0	0	...	0	0	0	0
2	1	1	0	0	...	0	0	0	0
3	1	0	0	0	...	0	1	0	0
4	1	1	0	0	...	0	0	0	0

Figure 5.1: One-hot encoding on genres

5.1.3 Word Embeddings

Next, we explored a method for comparing plots. As an extension to one-hot encoding, we used word embeddings to translate the plot for each movie. The goal of using word embeddings is to connect words that often occurs in the same context and are "similar" to each other. It allows the words to be represented as a dense vector. However, one-hot encoding was kept for directors, actors, and genres, since the differences of the plot recommendations were the most interesting to evaluate further. We tried out two different approaches to word embeddings that are described below.

GloVe. To create a vector representation of the plot description of the movies, we used the unsupervised learning algorithm GloVe. The GloVe algorithm has been pre-trained on word to word co-occurrence statistics from a corpus, which shows how frequently words co-occur with one another and gives a representation with linear substructures of the word vector space. The intuition is that the ratios of word to word co-occurrence probabilities have some kind of meaning of how the words can be encoded. We applied these pre-trained word vectors and used them on the plot descriptions to be able to compare the similarity between the plots (Pennington et al., 2014).

Sentence-BERT. S-BERT is a technique developed for complete sentences instead of individual words. It is a modification of the pre-trained Bidirectional Encoder Representations from Transformers (BERT), which have learned contextual relations between words in a text. However, S-BERT is optimized for sentences and has obtained better results compared to other methods. The initial approach, BERT, proved slowly when computing similarity between longer texts, and S-BERT was chosen due its performance and speed.

Reimers and Gurevych (2019) described how the same computational task can be reduced from 65 hours with BERT to 5 seconds with S-BERT. The model reached this improvement in speed by creating Siamese and triplet networks, which makes the sentence embeddings comparable with cosine similarity. S-BERT has been trained on multiple datasets and tested on multiple common semantic textual similarity tasks and shows improvement over multiple previous techniques, such as average BERT embeddings and average GloVE embeddings, which was used in the previous approach.

5.2 Association rule mining

Association rule mining, also called rule-based collaborative filtering or market basket analysis, was first introduced for supermarket data to predict which items are frequently bought together. This enables marketers to give more accurate advertising and placement in a store.

Market basket analysis uses data over multiple transactions, in this case movies and other content watched during a flight, previously described as a session. The transactions contains multiple items bought or watched together, and these can be used to find patterns over frequent items with positive or negative dependence to each other. A rule is of the format:

$$X \rightarrow Y,$$

where X (also called the **antecedent item**) and Y (also called the **consequent item**) is one or more items as for example:

{Love Actually, About Time} → {Bridget Jones's Diary}.

Multiple factors are relevant in calculating and comparing the most relevant rules (Tan et al., 2018):

1. **Support** is the fraction of the sets which contains both X and Y, which can be set to a minimum to limit the number of calculations. See Equation 5.3.

$$Support(X) = \frac{|t \epsilon T; X \subseteq t|}{|T|} \tag{5.3}$$

2. **Confidence** is how often Y appears in sets with X and can be used to sort the results in descending order to get the most confident rules first. See Equation 5.4.

$$Confidence(X \Rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)} \tag{5.4}$$

3. **Lift** is used to measure the independence between X and Y. If the lift is equal to 1, X and Y are independent from each other. A lift > 1 indicates dependence while a lift < 1 indicates negative dependence. See Equation 5.5.

$$Lift(X) = \frac{Support(X \cup Y)}{Support(X) \times Support(Y)} \tag{5.5}$$

4. **Conviction** describes the frequency where X occurs without Y, and can be used to measure how often a prediction would be wrong. See Equation 5.6.

$$Conviction(X \Rightarrow Y) = \frac{1 - Support(Y)}{1 - Confidence(X \Rightarrow Y)} \tag{5.6}$$

5. **Leverage** is a comparison of the frequency of X and Y together and if they were statistically independent. It results in a value between -1 and 1, where 0 indicates independence. See Equation 5.7.

$$Leverage(X \Rightarrow Y) = Support(X \cap Y) - Support(X) \times Support(Y) \tag{5.7}$$

To limit the necessary calculations, we can set a threshold called minimum support, `min_support`, which is defined as follows for a transaction base T, where $T = \{T_1, ..., T_m\}$. The support of an itemset $X \subseteq I$ is the fraction of transactions in T, of which X is a subset. All itemsets with higher support than minimum support are considered frequent itemsets (Aggarwal, 2016).

5.2.1 Apriori

Apriori is one of the earliest algorithms for frequent pattern mining and market basket analysis, and can be used to create association rules. It uses joins to generate candidates from frequent patterns appearing in the analyzed transactions. It is efficient due to the fact that every subset of a generated frequent pattern is also considered frequent, and also by joining two itemsets with $k-1$ items in common. E.g.

$$\{i_1, i_2, i_3\} \text{ and } \{i_1, i_2, i_4\}$$

have two items in common and can together create

$$\{i_1, i_2, i_3, i_4\},$$

which will be considered a candidate as it might be frequent compared to the minimum support. Another reason to the efficiency of the algorithm is the pruning of subsets. If

$$\{i_1, i_2, i_3\}$$

is not considered frequent, then

$$\{i_1, i_2, i_3, i_4\}$$

can be pruned without further computations.

Figure 5.2 shows the pseudo code for a recursive Apriori algorithm which implements the above mentioned techniques (Aggarwal and Han, 2014).

Algorithm *Apriori*(Database: $\mathcal{T}$, Support: s)
begin
  Generate frequent 1-patterns and 2-patterns
    using specialized counting methods and
    denote by $\mathcal{F}_1$ and $\mathcal{F}_2$;
  $k := 2$;
  while $\mathcal{F}_k$ is not empty **do**
  begin
    Generate $\mathcal{C}_{k+1}$ by using joins on $\mathcal{F}_k$;
    Prune $\mathcal{C}_{k+1}$ with *Apriori* subset pruning trick;
    Generate $\mathcal{F}_{k+1}$ by counting candidates in
      $\mathcal{C}_{k+1}$ with respect to $\mathcal{T}$ at support s;
    $k := k + 1$;
  end
  return $\cup_{i=1}^{k} \mathcal{F}_i$;
end

Figure 5.2: Apriori algorithm

Implementation

We created the association rules using the Apriori algorithm in multiple experiments below. During these experiments, we altered different factors to compare the different results, such as `min_support`, `field of interest`, `ascendant item`, and also `size of dataset`. We tried two different Python libraries with support for Apriori: apyori and mlxtend, but

we found Apyori to be too slow to work with larger datasets, so we calculated all numbers below using mlxtend (Raschka, 2018).

We computed the accuracy by comparing a given recommendation to the last watched item by the passenger, which is further described in Section 6.2. We considered every rule with lift above 1 valuable for the recommendations since this indicates a dependency between the items, and we sorted the results out in descending order after confidence.

We varied the minimum support to see differences in accuracy, but since a lower minimum support creates more number of rules we preferred to keep this value considerably low, to be able to cover more categories with the created rules.

Figure 5.3 shows an example of how the rules look during an experiment. The columns and their meaning are explained in Equations 5.3–5.7.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
114	(cartes et camera)	(Playlists)	0.005469	0.086896	0.005417	0.990431	11.397931	0.004942	95.419405
117	(la collection)	(Playlists)	0.002041	0.086896	0.002002	0.980769	11.286747	0.001825	47.481426
84	(Movie)	(Sit back)	0.002931	0.051174	0.002852	0.973214	19.017617	0.002702	35.422824
118	(nos sorties)	(Playlists)	0.003808	0.086896	0.003638	0.955326	10.993950	0.003307	20.439490
81	(Latest Releases)	(Sit back)	0.003651	0.051174	0.003376	0.924731	18.070207	0.003189	12.605826
120	(toujours à l'affiche)	(Playlists)	0.002748	0.086896	0.002538	0.923810	10.631252	0.002300	11.984495
67	(Help Video)	(Language Screen (destination))	0.001937	0.002852	0.001753	0.905405	317.411046	0.001748	10.541274
125	(Western Cinema)	(Sit back)	0.008139	0.051174	0.005757	0.707395	13.823242	0.005341	3.242690
112	(albums)	(Playlists)	0.002421	0.086896	0.001649	0.681081	7.837919	0.001438	2.863124
66	(Language Screen (destination))	(Help Video)	0.002852	0.001937	0.001753	0.614679	317.411046	0.001748	2.590212
33	(Chinese Cinema)	(Sit back)	0.002460	0.051174	0.001387	0.563830	11.017819	0.001261	2.175356
78	(The Collection)	(Latest)	0.006425	0.043101	0.003598	0.560081	12.994604	0.003321	2.175173
30	(Binge-watching)	(Latest)	0.002905	0.043101	0.001544	0.531532	12.332209	0.001419	2.042611
126	(Route, Default)	(View All)	0.003009	0.010245	0.001557	0.517391	50.500167	0.001526	2.050843
38	(Collections)	(Latest)	0.004069	0.043101	0.002054	0.504823	11.712541	0.001879	1.932439
123	(View All)	(Route)	0.010245	0.016029	0.005077	0.495530	30.915005	0.004913	1.950505
29	(BA Recommends)	(New Releases)	0.004462	0.242277	0.002080	0.466276	1.924558	0.000999	1.419690
20	(All (A-Z))	(Latest)	0.005705	0.043101	0.002630	0.461009	10.696001	0.002384	1.775353
128	(Default, View All)	(Route)	0.003402	0.016029	0.001557	0.457692	28.554396	0.001503	1.814415
22	(All contents)	(All Contents)	0.002813	0.006268	0.001282	0.455814	72.725640	0.001265	1.826089
12	(Home)	(All)	0.004868	0.043310	0.002107	0.432796	9.992874	0.001896	1.686675
70	(Hollywood and More)	(Welcome)	0.007445	0.017991	0.003206	0.430580	23.932417	0.003072	1.724577
74	(KLM 100 Years)	(Latest)	0.006922	0.043101	0.002970	0.429112	9.955935	0.002672	1.676157

Figure 5.3: An example of the sorted Apriori rules

5.3 Combination of models

In the last step, we combined the two approaches, content-based and market basket analysis, into one final model. We chose the movie recommendation model with the best feedback from our evaluation tests, further described in Chapter 6, and the Apriori model with the best overall accuracy. In this combined model, two categories were first recommended by the Apriori model, based on the watched item. A check was performed to see if any of these two categories existed on the current flight, since the categories varied depending on the flight.

Secondly, a list of all the content during that flight was collected. If the recommended categories were available on that flight, the list would be created from the content in those specific categories. However, if they were not available on this flight, the list would then be consisting of all content on that flight instead. If the previously watched item was a movie, TV show or music, the next step was to apply this item into the content-based model. By using the TMDb and Spotify Web API, as described earlier, information about all the content could be fetched and applied to the content-based model. It was then possible to get a list of the recommended movies and TV shows or music available on that flight for that watched item. These two recommendations, the two categories and the specific items, could then both be presented to the passenger.

5.3.1 Adding model for TV shows

When integrating the TMDb API into our models, data for TV shows also became easily available. We structured this data in the same way as the movies, with the same fields, i.e. actors, directors, genres and overview. This enabled TV shows to simply be added to the already existing movie model, and recommendations for both types could be interchangeable.

5.3.2 Adding model for music

When combining the models, we found it desirable to complement the models with recommendations for music as well since this was also shown to be a popular category. Therefore, we used the public Spotify Web API to fetch information about each track. We then applied the information to the models that were already created for the movie recommendation. However, this model would only compare the genres of the artist or track, and not other attributes. We therefore handled music separately, since this data did not include the same information as the movies and the TV shows, and the genres of music and movies were different.

Chapter 6
Evaluation

6.1 Content-based recommendation

To evaluate the content-based recommendations, we used two different evaluation methods. The two different approaches are presented and described below.

6.1.1 Initial evaluation

A difficulty with evaluating recommendation systems is that it is hard to know if a recommendation is good or bad without ratings. Therefore, we made a fast and simple initial evaluation by taking inspiration from Tang et al. (2020), the creator of the Weighted-PathSim. In this approach, we used a movie with sequels in the models to see that the movie sequels appeared in the recommendations, for example *Harry Potter*. This means that when creating recommendations for a Harry Potter movie, preferably the other Harry Potter movies would be included in the recommendations. However, a more accurate approach of evaluation is possible to assess by a click-through experiment, which was carried out later during the project.

6.1.2 Model evaluation

Next, we evaluated the three content-based models using an interface and ten test users to get a click-through rate (CTR) to be comparable to other models for further evaluation.

We created a simple interface using the standard GUI Python package Tkinter. The interface consisted of a drop down menu, where the test user could choose between ten different movies, from different genres. After selecting a movie, the user was presented with ten suggested recommendations of new movies, with associated keywords and image of the movie. The test user could then, for each of the suggested movies, decide if they would consider to

watch it, by choosing *yes*, *no* or *maybe*. The user's response was registered and saved in a separate file. This procedure was then repeated for all three of the models using the same movie, in order to compare the results of how good the recommendations were for that movie.

The test was performed twice on the same test user but with two different movies. The reason for this was to avoid results too heavily reliant on the recommendations of individual movies.

The points were distributed so that a *yes* gave 1 point, a *maybe* 0.5 points and 0 points for a *no*. The argument for not giving -1 point for a *no* was that a *no* doesn't necessarily mean that the recommendation is bad, and therefore should not be punished with deducted points.

To measure the accuracy, we computed the mean value for each model with a maximum score of 10 and a minimum value of 0. Each test was performed in a random order to avoid biased results as much as possible. Figure 6.1 shows the test interface.

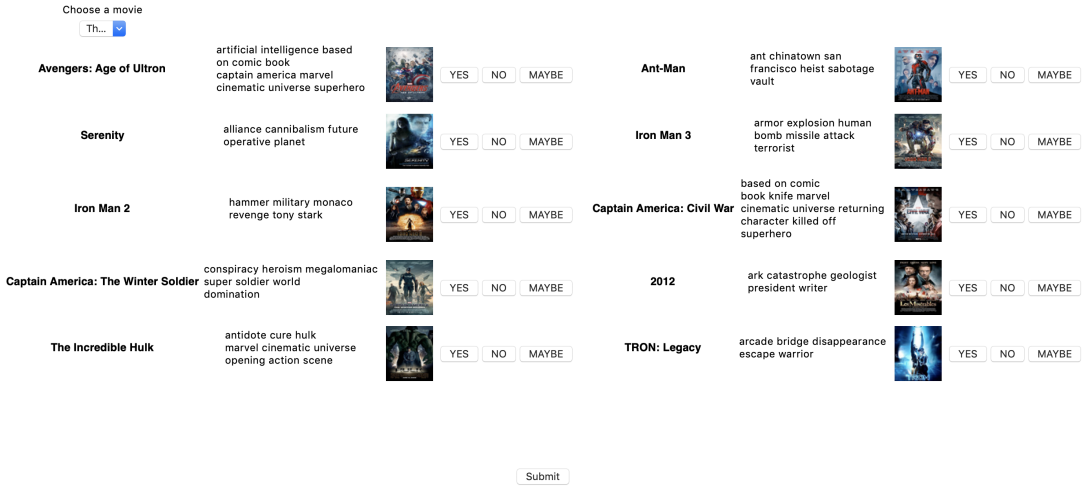


Figure 6.1: The user test interface showing the recommendations for the movie *The Avengers*.

6.2 Association rule mining

We calculated the accuracy of the association rule mining models by comparing the recommended item to the last item watched by the passenger. A correct prediction corresponds to when these two items match. All initial calculations were done on the same set of data, with a train set size of 80% and test set size of 20% of the total data, randomly selected for each run. The data was sorted and grouped by each passenger first after the data was divided. We used the train set to create the rules, and the test set to evaluate the created rules.

As mentioned earlier, we removed duplicates from the same passenger, since for example a paused and played movie, will show as two separate items and therefore affect the results. As also mentioned earlier, the main parameters that were varied during the different experiments were the `min_support`, representing how big percentage of the users that watched the included categories, and the `antecedent item`, which item the recommendation was created from. This item could either be the second to last watched item [-2] or the first watched

item [0].

To not make the result dependent on only one dataset, we performed the experiments on several datasets. All these experiments were made with randomised datasets with one million entries, to compare and make sure that the numbers turned out fairly similar. We also made experiments with larger datasets, since an increase in data usually leads to better results. However, for these experiments, the `min_support` could not be set to the desired level, and instead had to be higher. Finally, another attempt was made where we instead beforehand filtered out every data item without any `category_title`, which would result in a dataset where all data points could be trained on. However, this did not make any significant difference in accuracy compared to the first dataset.

When comparing two datasets, both containing one million entries, there were quite a big difference in the number of individual passengers. Therefore, we chose the dataset with the highest total number of passengers, since this dataset gave the most variation and created the highest number of Apriori rules.

6.3 Combined model

The final model is a combination of the previous models with the best obtained results. To evaluate the combined model, we used a similar approach as for the Apriori model evaluation, i.e. comparing the recommended movie/s to the actual last watched movie. The rules for category recommendations are created from the total training data. However, this evaluation method could only be done on a limited part of the test data since the evalutaion required a lot of computation. Therefore, a limited, random slice of the test data was used to run the evaluation tests. We decided to use a test set size of 100, 500 and 1000 data points.

Chapter 7
Results

In this chapter, we will present the findings from the content-based recommendations, association rule mining recommendations and the final combined model. These findings are retrieved from the evaluation methods described in the previous chapter.

7.1 Content-based recommendation

The results from the evaluation tests can be seen in Table 7.1.

model	mean accuracy
One-hot encoding	66.00 %
GloVe	67.75 %
S-Bert	63.23 %

Table 7.1: Results from the evaluation tests of the content-based models

7.2 Association rule mining

In this section, we present the results from two baseline models as well as the results from our Apriori models.

7.2.1 Baseline model

To be able to compare our results from the different models, two versions of a baseline model were created. The first baseline model recommends the most popular `category_title`

of the current dataset to everyone, then comparing this to the last item watched by each passenger. This model resulted in an accuracy of 0.18076, ~18%.

The second baseline model recommends the most popular `category_title` of each flight to passengers on that particular flight. This resulted in an accuracy of ~43% on category recommendations. This was however decided to be an unfair comparison because of two main reasons:

1. It is impossible to create rules or models for individual flights in the following experiments due to very limited data.

2. The information about the most popular content for each flight will not be available during the flight, only after. It will in practise be impossible to implement.

7.2.2 Apriori results

The results from the evaluation of the Apriori models are collected in the tables below. The tables present a few different values, which are explained further below.

- **min_support** – the minimum support, see Equation 5.3.
- **antecedent item** – indicates if the recommendations were based on of the first item watched `[0]` or the second to last item watched `[-2]`.
- **correct predictions** – the number of correct predictions.
- **total checked** – the number of rules the model was able to create and compare, i.e. when a rule could be found for the antecedent item. Varies mainly due to the minimum support, since not all items pass this limit which results in fewer rules.
- **all** – the total number of items passed through to try to make recommendations.
- **accuracy (total checked)** – number of correct predictions out of the rules that were possible to make.
- **accuracy (all)** – the total number of correct predictions out of all items tried.

With a `min_support` of 0.001, 166 rules were created for 1114 categories, which indicates that many categories were not covered during the recommendations. However, when `min_support` was decreased to 0.0005, the number of rules went up to 474.

Since we discovered an increase in accuracy when the `min_support` was lowered, seen in Table 7.2, we experimented with even lower `min_support`. However, due to limited processing power, a support lower than 0.00025 for one million data points was not possible with the tools available. After reaching this limit, we made an attempt to compute this using AWS Sagemaker instead. Due to the limited RAM of the available AWS instance, this did not give any better results, and that is the reason why no result of lower minimum support has been calculated.

We could see small increases in accuracy due to changes of different factors during the first experiments. These experiments include;

1. Removing duplicates before collecting the last watched item;
2. Running on different sizes of datasets;
3. Running on different filtered datasets, for example where `category_title` ≠ null.

All tables below are created after these initial experiments, and are therefore run with the same foundation and instead varying other factors such as `min_support` etc.

Tables 7.2 and 7.3 show the results from the different runs of the Apriori model, where one category has been recommended and compared to the actual last watched category. It can be seen that the accuracy is improved as the `min_support` decreases, as well as the number of total checked rules, which indicates that more rules are created. This is desirable as a model with fewer rules would not be able to give recommendations to all antecedents. The accuracy of `category_parent_title`, Table 7.3, is slightly higher compared to `category_title`, Table 7.2, which was expected due to fewer and broader categories.

Tables 7.4 and 7.5 show the results from tests with recommending two or three top categories. These experiments were done only on `category_title` and not on any higher level categories. We decided that the recommendations would be more useful for `category_title` compared to `category_parent_title`, which only included less specific categories such as *Movies* and similar.

min_support	antecedent item	correct predictions	total checked	all	accuracy (total checked)	accuracy (all)
0.01	[0]	279	10569	43369	2.640 %	0.643 %
0.01	[-2]	299	1299	43301	23.02 %	0.691 %
0.001	[0]	1539	32412	43335	4.748 %	3.551 %
0.001	[-2]	1536	5251	43126	**29.25** %	3.562 %
0.0005	[0]	1838	36058	43359	5.097 %	4.239 %
0.0005	[-2]	1744	6198	43296	28.14 %	4.028 %
0.00025	[0]	1982	38929	43252	5.091 %	**4.582** %
0.00025	[-2]	1894	7114	43177	26.62 %	4.387 %

Table 7.2: Apriori findings for category_title

min_support	antecedent item	correct predictions	total checked	all	accuracy (total checked)	accuracy (all)
0.01	[0]	1285	23123	43009	5.557 %	2.988 %
0.01	[-2]	1339	2941	42966	**45.53 %**	3.116 %
0.001	[0]	2568	40512	42975	6.339 %	5.976 %
0.001	[-2]	2581	5953	43151	43.36 %	5.981 %
0.0005	[0]	2695	41122	42826	6.554 %	6.293 %
0.0005	[-2]	2617	6103	42791	42.88 %	6.116 %
0.00025	[0]	2683	42231	42817	6.353 %	6.266 %
0.00025	[-2]	2706	6351	42828	42.61 %	**6.318** %

Table 7.3: Apriori findings for category_parent_title

min_support	antecedent item	correct predictions	total checked	all	accuracy (total checked)	accuracy (all)
0.01	[0]	376	10590	43160	3.551 %	0.871 %
0.01	[-2]	375	1322	43182	28.37 %	0.868 %
0.001	[0]	1861	32086	43143	5.800 %	4.314 %
0.001	[-2]	1926	5252	43269	36.67 %	4.451 %
0.0005	[0]	2238	36008	43288	6.215 %	5.170 %
0.0005	[-2]	2230	6321	43516	35.28 %	5.125 %
0.00025	[0]	2675	39007	43140	6.878 %	**6.201 %**
0.00025	[-2]	2641	7190	43255	**36.73 %**	6.106 %

Table 7.4: Apriori findings for two recommended categories

min_support	antecedent item	correct predictions	total checked	all	accuracy (total checked)	accuracy (all)
0.01	[0]	386	10533	43375	3.665 %	0.890 %
0.01	[-2]	367	1313	43007	27.95 %	0.853 %
0.001	[0]	2097	32016	43091	6.550 %	4.866 %
0.001	[-2]	2082	5173	43295	40.25 %	4.809 %
0.0005	[0]	2550	35878	43276	7.107 %	5.892 %
0.0005	[-2]	2569	6315	43310	40.68 %	5.932 %
0.00025	[0]	3037	38789	43278	7.830 %	**7.017** %
0.00025	[-2]	2971	7225	43098	**41.12 %**	6.893 %

Table 7.5: Apriori findings for three recommended categories

7.3 Combined model

The results from the combined model are presented in Table 7.6. A correct prediction corresponds to if any of the three recommended items are a match with the last watched item. In this model we used both the movie recommendation model with GloVe embeddings, as well as the Apriori model with `min_support` 0.00025 and two recommended `category_title`. To detect possible irregularities in the results, we ran this evaluation a couple of times.

number of items	min_support	checked items	correct predictions	accuracy (total checked)	accuracy (all)
100	0.00025	54	5	9.259 %	5.000 %
500	0.00025	279	60	21.51 %	12.00 %
1000	0.00025	515	8	1.553 %	0.800 %

Table 7.6: Results for the combined model

The structure and columns of the table are as followed;

- **number of items** – the total size of test set.
- **min_support** – the minimum support, see Equation 5.3.
- **checked items** – the number of movies, TV shows and songs the model was able to make recommendations for.
- **correct predictions** – the number of correct recommendations compared to the actual last item watched.

- **accuracy (total checked)** - the number of correct predictions out of the item recommendations that were possible to make.
- **accuracy (all)** - the number of correct predictions out of all items in the test set.

Chapter 8

Application

Figure 8.1 shows the original application interface. It has a general start screen with the less specific categories, such as *Movies*. When entering *Movies*, the user will see a slide menu of movie categories such as *Action*, *New releases*, etc. When a movie ends, the passenger will be taken back to the start screen shown in Figure 8.1.

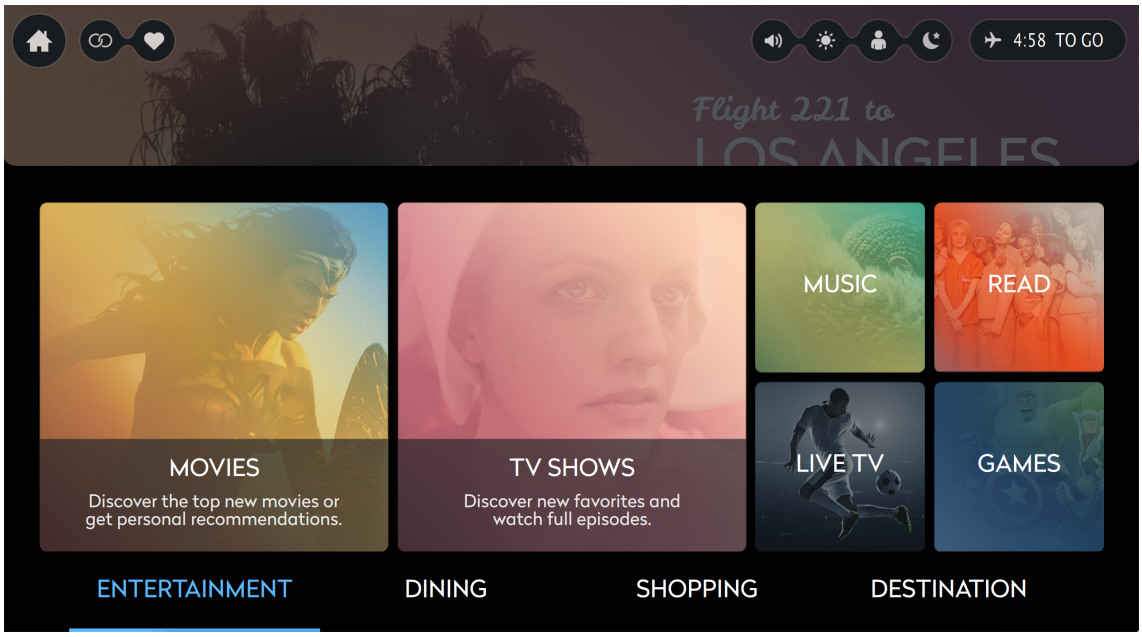


Figure 8.1: The original IFE screen

In the section below, we propose a few suggestions on how the recommendations could be integrated into the existing system.

8.1 Interface suggestions

The final model was not integrated into an existing IFE system due to limited time and limited access to the necessary tools, but below are some suggestions on how the recommendations could be presented.

Figure 8.2 shows the IFE start page with two recommended low level categories in addition to the standard higher level categories. Due to the functionality of our model, this requires at least one previous interaction with the IFE, which could be a movie, TV show or a song.

Figure 8.3 shows a suggested implementation of an explicit feedback system, where this screen appears after finishing watching a movie. The user is given the option to rate the movie as a similar movie would not be desired to recommend if the passenger did not like the first movie. If the user chooses thumbs up, the two most similar movies according to the content-based model available on the current flight are recommended, which can be seen in Figure 8.4.

Another suggestion is to rank all the movies in an entered category according to similarity from the previous watched movie. This is not shown in the interface suggestions, but could be handled by our model which already calculates similarity for all movies on the same flight, and these could be sorted into descending order within each category.

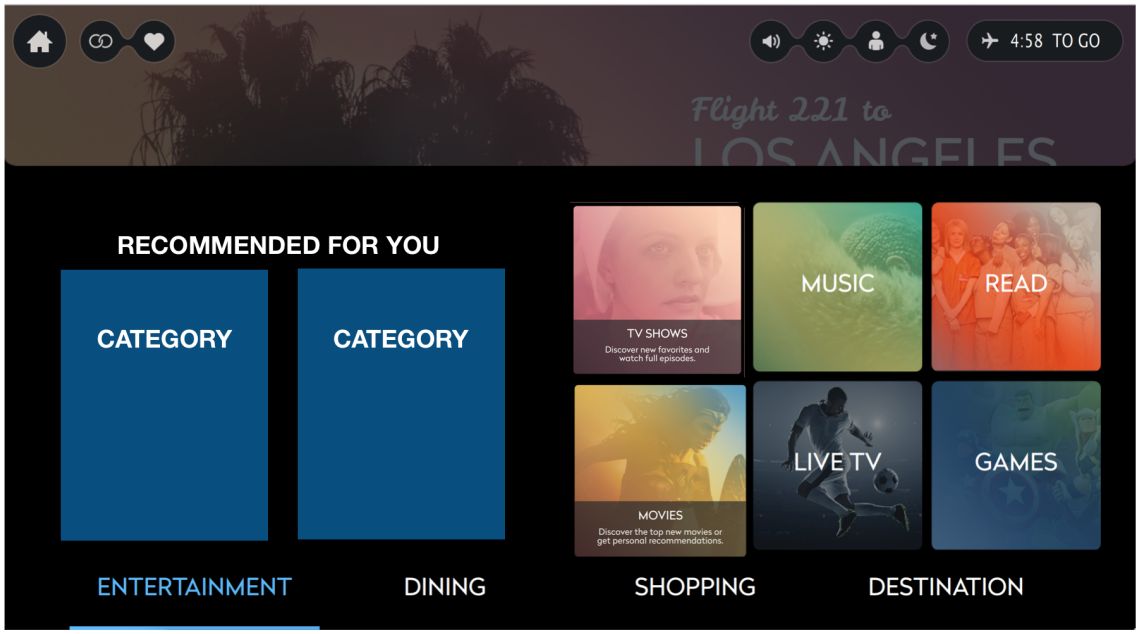


Figure 8.2: IFE startpage with two personalized recommendations of categories

Figure 8.3: Rating option after finishing watching a movie

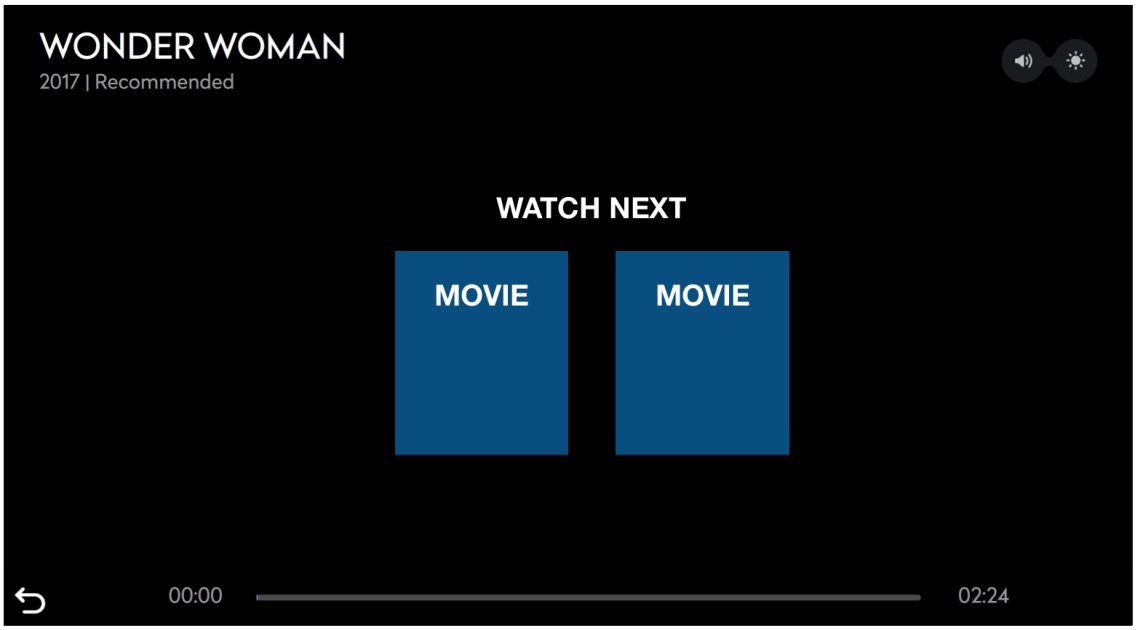


Figure 8.4: Recommended movies after finishing watching a movie

Chapter 9
Discussion

Most of the previous personalization attempts for in-flight entertainment systems have focused on pairing the IFE with various applications or social media to find the demographics to be able to group similar passengers together. Therefore, after discussion with Tactel, we decided to focus on the passengers who choose not to connect any devices or social media to the IFE, the so called *cold-start*.

We wanted to find a suitable solution for these users as well, even if the recommendations could not reach the same level of personalization as for a logged-in user. We also wanted to steer the focus away from demographics as these often become stereotypical.

9.1 Comparison of content-based recommendation models

The mean values from the evaluation tests showed that model using GloVe for word embeddings on the plot gave the most reliable recommendations, since the suggested movies were considered to be better recommendations by the test users. This is the reason this model was used in the final combined model as well. However, the results from the three different models were very similar and it only differed a very small percentage between them.

We expected S-BERT to be the best model for the content-based recommendations, due to S-BERT being compared to GloVe during the creation of S-BERT and proving better at these tests, as seen in Reimers and Gurevych (2019). When comparing only plot descriptions directly to each other during implementation, we perceived a strong similarity almost immediately using S-BERT, but during the evaluation tests the model using GloVe embeddings performed the best. To be taken into account, our tests were very limited, but they also did not only test the plot embeddings, but rather a combined result. This could explain this result. A conclusion could also be that the genres, actors and directors outweighs the plot when choosing a movie.

In retrospect, we also discovered that sometimes the test user had a hard time knowing whether or not they would consider watching the recommended movies. This could have been improved by adding more information about the movies in the test interface, for example which actors were starring in the movie or more focus on the movie posters.

9.2 Analysis of Apriori model

When evaluating the Apriori model, we chose to look at two different calculations of accuracy. The reason for this was to show how many failed recommendations were due to no existing rules rather than a false prediction. First, we looked at the total number of correct recommendations for all items. This resulted in a top accuracy of ~6-7 % for our best model. This did unfortunately not exceed the baseline model which recommended the most popular category to everyone. Next, we looked at the number of correct recommendations for the items that we were able to create rules for, which resulted in an accuracy of ~30-40 %. This means that about one third of the time that we manage to create a rule, it was correct. This result exceeded the baseline model. Considering the large differences in accuracy, it might be a good idea to focus on giving more accurate recommendations only when it is possible.

Antecedent item. When alternating between creating the rules based on the first item or second to last watched item, it was found that the accuracy was in most cases higher when using the second to last item. We believed the reason for this was that the second to last item and the last item, that we are trying to predict, are closer to each other in time than the first item, since they are watched after each other. Therefore it is more likely that they are more similar. It was therefore decided to use the second to last watched item for the final combined model.

Number of categories. We also found that the recommendation of three categories, compared to one or two, resulted in a slight improvement in accuracy. This was probably due to that the probability of recommending the right category is higher if we recommend more categories. However, to make the recommendations more specific, we decided to use the model that recommended two categories. This was also partly a design choice, as three recommendations looked more cluttered in the interface.

Level of recommendation. We made recommendations for `category_title`, (lowest level category), and `category_parent_title`, (mid level category). The results showed that the recommendations using `category_parent_title` had higher accuracy. However, we believe that the reason for this is due to the categories are less specific, such as 'Movies' instead of 'Action'. Therefore, `category_title` was used in the final model since it was desired to have more specific recommendations. On the other hand, `category_parent_title` could act as a substitute when a `category_title` recommendation can't be done, since the total number of items covered is higher. We also tried to make Apriori recommendations using `media_title`, but due to the much higher amount of different items, this was not successful.

Minimum support. Comparing the different models, the results proved that when using more data `min_support` had to be increased to be able to run on our computers. However, when increasing the `min_support`, less categories met the requirements to be part of the rules. In most cases, this results in a lower total accuracy and therefore it was found that the best approach was to use one million data entries and a `min_support` of 0.00025. However, in Table 7.3, we can see an increase in the total checked accuracy when the `min_support` was higher. In these cases, we have a higher percentage correct recommendations, but the variation among the recommended categories will be very low. We strive to recommend many different categories and therefore, we decided that a lower `min_support` would give us more value.

In conclusion, we believe that the Apriori model is a good approach for creating personalized recommendations in the IFE. Using patterns in other passengers previous behavior to create rules for recommendation, is shown in this project to give useful results.

9.3 Combination of models

To be able to provide more specific and interesting recommendations, we combined the two models, content-based and market basket analysis, into one final model. This was carried out by sorting the movies in the recommended categories according to similarity to the previously watched content.

To calculate the accuracy, we used the same approach as for the Apriori models, by comparing the recommended item to the last watched item. The same problem as in previous attempts still stands, i.e. a recommendation could still be appreciated by the passenger, and the result does not necessary reflect the real-life experience of the users. As seen in Table 7.6, the highest accuracy obtained was 12 % for the overall accuracy and 21.51 % for the total checked. However, this test was performed with a test set of only 500 data points, and therefore one correct prediction corresponds to a high percentage of the total items. We did expect low results, since the probability of a match for a predicted media item was lower due to a lot more items compared to a category recommendations as in previous models. Due to the long execution time, the tests are very limited in size, and we feel that the tests preferably should be run on more data before being to heavily relied on.

Even with slightly incomplete test results, we argue that a combination of the models still is an improvement, since it is merely an extension of the category recommendations. Now we are also able to rank the movies within each category in descending order according to similarity, as well as give two movie recommendations after a previous movie has been watched, as portrayed in Figure 8.4.

9.4 Limitations

One of the limitations of this project was that we did not have access to the real user data from the in-flight systems until halfway through the project timeline. This was the main reason why we started with public data sets and focusing on movie recommendation models initially.

If we would have had access to the real data earlier, more discoveries related to data mining may have been found. We were aware of this problem from the start and therefore planned for the uncertainty of not knowing when we would receive the data. However, the goal would still be a hybrid model, but we might have started in the opposite end, first with market basket analysis and then creating the content-based recommendations. So in the end, the results would probably be similar to the results of our current approach.

When we received access to the real user data from the IFE systems, we could start exploring and preparing the data. During this phase, we discovered other limitations. Firstly, there were no recorded data of in-flight shopping or any meal orders and similar, which were part of our initial plan to include. But unfortunately, this had to be skipped completely since the lack of information made it impossible. Secondly, a dataset of each flights total content selection was also not available, and therefore content that no passenger had interacted with will not be covered by our models.

9.5 Future work

This subject provides a lot of possibilities of further experimentation and some of them will be discussed below.

9.5.1 Testing

We believe that the best way of evaluating recommendation systems would be to let people test the system themselves, preferably in a live A/B testing method. This would give the opportunity to compare a personalized system with recommendations to a non-personalized system. This was not available to us during the thesis work mainly due to security reasons and long deployment phase. Since there is no easy way to measure how good a recommendation actually is, this would provide a more accurate result that shows whether the recommendations are used by the passengers or not.

9.5.2 Model improvement

To improve the content-based recommendations, actor embeddings could be added. We decided not to do this, since it would require too much time to implement, and it was not the main focus of our project.

The decision to not explore the solution with a login was taken since it already exists solutions with that approach. However, this created a cold-start problem. To make the first recommendations even better, and limit the cold-start problem, a few initial questions could be asked to put the passenger in a certain category, as proposed by Shi et al. (2017). We did not test this approach at all, since that type of data could not be retrieved or would require excessive user testing, and similar solutions already exists. Instead we wanted to explore personalization for customers that did not pair a personal device or create a login.

Regarding improvements for the model we did implement, one improvement would be to train the model as more content is watched, and make better recommendations further into the flight. Another improvement could be to let the user rate the movie etc. after watching. Right now, we are only assuming that the user likes the movie if s/he have watched it, but

this does not need to be the case. Adding the option of rating the product could give the model information about whether the user liked this item or not, and could then instead recommend something different if the user did not like the item. A proposed solution of this can be seen in Figure 8.3.

An alternative to ratings could be to take the duration of the content watched into account when creating the recommendations. If a passenger only watched a movie for a short period of time, this type of movie might not be of interest for the passenger and therefore similar movies should perhaps not be recommended.

9.5.3 Additional areas

From the beginning, the plan was to include in-flight shopping, meals, beverages and other available products to achieve a broader perspective of each passengers flight and timeline. However, there was no collected data of this, which made it an impossible addition to our models. This would be an interesting area to explore if this type of data becomes available.

One future goal would also be to make the recommendations more integrated with each other. As of now, the model gives movie and music recommendations separately, but we believe with more data and training, it would be possible to incorporate all areas with each other. Hopefully, some interesting suggestions can then be seen, for example a certain drink combined with a specific movie.

Chapter 10
Conclusion

Most previous approaches to personalize IFE systems have focused on logging on to different social media, creating an account or downloading an application before boarding. We feel that this thesis provides a solution for the passengers who are not willing to take these initial steps. Independently of the results, this is an area worth covering and further explore as it could lead to more satisfied passengers and higher profit for the airlines.

How can an IFE be personalized during flight with limited information? This thesis focused on solving the *cold-start* problem that arise due to limited information about the user. This was dealt with by two different approaches. By comparing the content of the items we could create content-based recommendations. These recommendations only compare the similarities of the items, e.g. the actors in movies, and can in that way provide recommendations without any information about the user other than the previous item watched. Another approach explored was market basket analysis, which compares different users behavior to each other. We created rules based on previous passengers behavior, which detect frequent patterns among the user interactions. These two techniques were found to be a good approach for giving recommendations after one media item had been watched by the passenger, which we think is a good solution for a system with limited information.

How effective are the different approaches and can they be combined? The obtained results did not exceed the results from our baseline model, where the most popular movie was recommended to all passengers. On the other hand, if only providing recommendations when it is possible, we obtain an improvement in accuracy compared to the baseline. This proves that our model is more effective when only giving recommendations we are certain of.

The models worked very well in combination, the association rule mining providing category recommendation and the content-based recommendations being able to compare similarities between the media items and giving recommendations based on this information.

What additional feedback system can be used to make the recommendations more accurate? There are many different techniques that can be used for creating valuable recommendations, and some have been discussed during this thesis. However, additionally to the approaches that were implemented, we believe that it would be complementary to add a feedback system such as ratings and few initial interview questions in order to receive more information about the user. By rating the previous watched item, we could detect whether the passenger liked the item or not. Giving recommendations similar to the first watched item would not be desirable if the passenger did not appreciate it. Initial interview questions could also be a good way to target the passengers preferences and some initial recommendations could be made even without any previous interaction with the IFE. In conclusion, we believe that one approach does not rule out the other and with a combination of different techniques, more accurate and appreciated recommendations can be provided.

References

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

Aggarwal, C. C. and Han, J. (2014). *Frequent Pattern Mining*. Springer Publishing Company, Incorporated.

Garcia, M. (2016a). Airlines target personalization with ife content recommendations.

Garcia, M. (2016b). Ife data informs airline content curation.

Hawk, E. (2018). Dynamic airline in-flight entertainment systems using predictive analysis.

Huang, A. (2008). Similarity measures for text document clustering. In *Proceedings of the 6th New Zealand Computer Science Research Student Conference*.

Jawaheer, G., Szomszor, M., and Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, page 47–51, New York, NY, USA. Association for Computing Machinery.

Kim, H., Katerenchuk, D., Billet, D., Huan, J., Park, H., and Li, B. (2019). Understanding actors and evaluating personae with gaussian embeddings. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

Koren, Y. (2009). The bellkor solution to the netflix grand prize.

Li, B. and Han, L. (2013). Distance weighted cosine similarity measure for text classification. In *Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning — IDEAL 2013 - Volume 8206*, IDEAL 2013, page 611–618, Berlin, Heidelberg. Springer-Verlag.

Liu, H. (2007). In-flight entertainment system: State of the art and research directions. In *Second International Workshop on Semantic Media Adaptation and Personalization (SMAP 2007)*, pages 241–244.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Piotte, M. and Chabbert, M. (2009). The pragmatic theory solution to the netflix grand prize, in: Netflix prize documentation.

Raschka, S. (2018). Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24).

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Researchscape International (2019). Trends in personalization. Technical report, Evergage.

Shi, L., Zhao, W. X., and Shen, Y.-D. (2017). Local representative-based matrix factorization for cold-start recommendation. *ACM Trans. Inf. Syst.*, 36(2).

Tan, P.-N., Steinbach, M., Karpatne, A., and Kumar, V. (2018). *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd edition.

Tang, Z., Yang, Y., and Bu, Y. (2020). Weighted-pathsim: Similarity measure for plot-based movie recommendation. Technical report, University of Illinois at Urbana-Champaign.

Töscher, A. and Jahrer, M. (2009). The bigchaos solution to the netflix grand prize.

Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

White, J. N. (2012). A history of inflight entertainment. *Design Affiliates, Inc.*

Wirth, R. and Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pages 29–39.

Zhang, X. and LeCun, Y. (2017). Which encoding is the best for text classification in chinese, english, japanese and korean?

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-11

EXAMENSARBETE Smart Personalization for In-flight Entertainment Systems
STUDENTER Sara Trygve, Frida Gunnarsson
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Jörn Janneck (LTH)

Smart personalisering av In-Flight Entertainment-system

POPULÄRVETENSKAPLIG SAMMANFATTNING **Sara Trygve, Frida Gunnarsson**

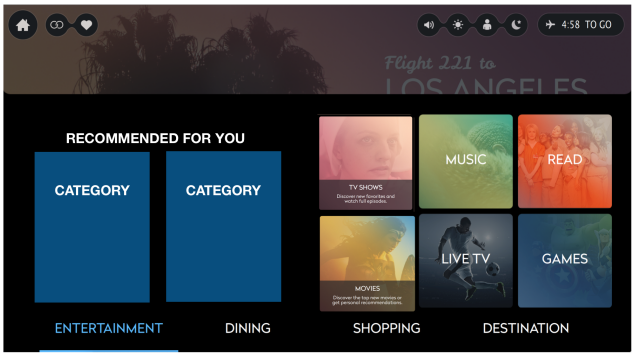
In-flight entertainment-system blir allt mer avancerade och större fokus på att ge varje passagerare en personlig upplevelse kan ses i branschen. Men hur ser möjligheterna ut för att skapa personliga rekommendationer utan inlogg och utan tidigare data?

Under långflygningar är en inbyggd skärm i sätet framför varje passagerare en allt vanligare syn, med innehåll så som shopping, karta över var du befinner dig och ett utbrett sortiment av media. Dessa kallas In-flight entertainment-system (IFE) och tillsammans med Tactel har vi undersökt möjligheter till att göra dessa mer personliga och kunna ge individuella rekommendationer.

En metod som tidigare använts för personalisering av IFE är en mobilapplikation som kan kopplas till IFE systemet och måste laddas ner i förväg. I vårt examensarbete har vi istället valt att fokusera på att skapa en personlig upplevelse för passagerare utan den möjligheten.

Detta har vi gjort genom att bygga ett hybrid lösning som kan appliceras efter att en passagerare har interagerat med minst ett objekt på skärmen. Rekommendationssystemet täcker både övergripande kategorier och specifika filmer, TV-serier och musik. Det utgår dels från likheten mellan olika objekt och dels från mönster i andra passagerares beteende. Likheten mellan bland annat filmer beräknas med hjälp av kosinuslikhet för skådespelare, regissörer och genrer, medan en jämförelse mellan filmernas handlingar görs med hjälp av två stycken word embedding-tekniker, GloVe och S-BERT.

För att dra nytta av data från tidigare passagerares beteende, har vi använt en metod som kallas Apriori. Genom att studera vad flera andra passagerare har tittat på under en flygning, upptäcks mönster mellan olika kategorier som finns ombord, och det skapas regler för vilka kategorier som ofta ses efter varandra. På så sätt kan en eller flera kategorier rekommenderas utifrån den förra kategorin som passageraren tittade på, samt att innehållet i kategorin kan rankas efter likhet med tidigare objekt.



För att mäta hur bra våra modeller har presterat, har vi jämfört vår rekommendation med det sista objektet som passageraren har interagerat med. Det är inte alltid vi kan skapa rekommendationer utifrån Aprioriregelrna, men när vi lyckas göra det uppnår vi en accuracy på cirka 30 % för kategorirekommendationer.