RECOMMENDATION SYSTEM AND USER INTERFACE DESIGN FOR EXPERIMENTAL FILM STREAMING SERVICE

RE:VOIR

AALBORG UNIVERSITET

Title Page

Title

Recommender System

And User Interface Design

For Experimental Film

Streaming Service

Company

Re:Voir

Theme

Master Thesis

Supervisor

Luis Emilio Bruni

Hendrik Purwins

Abstract

Re:Voir is an experimental film distribution company that wants to create subscription-based video streaming service. In this project, we find out what characterizes experimental films is and how they can be divided into categories. We explore state of the art literature about recommendation systems and analyse state of the art services such as Netflix and Youtube, and create a custom recommendation system tailored to experimental film. This project creates a pattern for implementation based on previous researchers. By trying to implement some of the generalised systems in the most adequate way. The main goal is to help Re:voir develop a recommender system corresponding to the research in this project. As Re:voir is a niche company the best way to achieve the goal is to follow the approach of the evolution of the internet from the ground up. This way they will acquire data the most efficient way possible and further build their data analysis. Lastly, we create a user interface design, based on current trends in this field.

Authors

Krišjānis Určs, Sevastian Tsvetkov

Table of Content

1 Introduction

"I live therefore I make films. I make films, therefore I live"
Jonas Mekas

With the introduction of the internet, people have repeatedly demonstrated their need for gaining new information and communication. These trends, managed to create a culture where we develop new ways for exchanging information, and the way it is expected to receive information. Web searches on a daily basis is a fitting example for such trends, as the amount of Internet users has increased dramatically over recent years. With the introduction to web services, such as YouTube where the information that is being uploaded has increased tremendously in the past couple of years. As huge amount of new information is transferred every day, it creates a problem. As the spread of information today is easier than ever, satiation increases, spreading the gap between what is out there and what we seek (Bawden D. 2009). In addition, people make use of web-based search engines, such as Google or Bing, searching for entertainment. However, the tremendous amount of information is often an obstacle to finding relevant information, created by the ease of sharing information with the internet.

In this context, not all data is reliable and valuable, which implies that it may become more and more difficult to obtain satisfactory results from just browsing the web. Iterating through searches often makes the flow if information slow and people often get lost finding the relevant information, then actually obtaining it. Several researches have been made to resolve such a problem with the use of recommendation systems (Bobadilla J., et.al. 2013). Users do not need to scan through all the search results to find what they need, instead the recommendation system filters through the results and presents the current most relevant information only in the results (Choi S., et.al. 2012). These systems are software tools and techniques providing suggestions for items that may be of use to a user. The suggestions relate to various decision making processes such as what items to buy, or what music to listen to online (Ricci F., et.al. 2011). Generally, the system`s design graphical user interface, core recommendation technique used to generate the recommendations are all implemented to provide useful and effective suggestions for a specific item. Items refers to the term used to denote what the system recommends to users.

2 Company Re:voir

Re:Voir is a company based in France that publishes and distributes DVDs of experimental and avant-garde cinema. Founded in 1994 to release VHS tapes of classic and contemporary experimental films, the line expanded to DVD in 2005, after a six-year period of researching proper compression techniques for these specialty films. Their film catalogue includes over seventy titles: films from the Dadaist, Surrealist and Leterist movements, films from the American avant-garde, diary films arthouse features, animated works and hand-painted films. They are devoted to give a wider audience access to a relatively unknown yet rich and diverse body of cinema and to disseminate the major works of experimental film at the highest possible quality. Each of their releases receives rigorous attention and

extensive preparation and yields an object of enduring integrity and quality for collection, discovery, and reflection *(REVOIR*, 2018).

Currently, the Re:voir video-on-demand project is to make available an important segment of their collection at the highest quality possible for digital reproductions. Most of the videos are originated from new 2K scans, tailor-compressed by Vimeo. Their current research initiatives include Graphic User Interface, with 3D navigation, visualisation relational database and the development of a Modular back-end office, that adds data, films filmmakers, tags and the relationships between them. In other words, the company want to expand towards digital media and offer their content as video-on-demand, where their audience would be able to view wide selection movies by monthly subscription, similar to that of Netflix, Hulu, HBO Now and others.

Their initiative towards a video on demand service has been enforced by researching current trends in the industry. Currently experimental cinema is a niche field, and the platforms for showcasing such art are sparse. Re:Voir is the only well known platform where experimental film authors can distribute their art. Hence the company is trying to expand its distribution, by modernising its platform to meet current tech trends. This will provide a better relationship between consumers and the company, and will expect to meet wider audiences.

Based on their needs, the company has pinpointed their objectives for this expansion. The main goal is to create a mobile application to integrate with their website (www.re-voir.com). This includes reworking their portal database for information regarding films and filmmakers, and determining needs for management and future development. Based on these needs the company will be able to push its subscription based VOD service to consumers.

3 Related studies

3.1 Experimental film

"On one hand, viewers goal is to integrate local detail into an appraisal of the films overall meaning and overal schematic structure. On the other hand, avant-garde film systematically confound the viewers effort to do so." (Peterson, 1994, p.28).

Avant-garde, independent, underground, personal, or experimental films - all of these are different names used for a film genre that rejects the conventional mainstream cinema and explores the possibilities of the medium itself by manipulating audiovisual elements in a novel and for many - strange ways. The historical avant-garde of the twenties was the first movement in art history that turned against the institution "art". Nevertheless, this film genre relates to art disciplines, such as literature, poetry, painting and dance. In this paper, we will use term experimental film, since it is the most widely used term and because it is possible to make experimental films without the presence of avant-garde movement in the cultural field.

Majority of experimental filmmakers begin as amateurs, experimenting with filmmaking technology available to them. Their aim is to render their personal visions, rather than to entertain or generate a revenue. Most of them conceive, shoot and edit their films alone or with a minimal crew. Furthermore, they often take the responsibility for financing production and distributing their films, which leads to total independence from anyone and allows to create films truly as visioned by the filmmaker. Because of small budgets majority of experimental films are short, often under thirty minutes. Nevertheless, there are some which also exceed mainstream cinema runtime and goes for eight hours and more. Many experimental films are silent, or use sound in non conventional manner. They generally avoid verbal communication, leading audience focus primarily on visual aspects of the film. (Sitney, 2002).

Experimental films seek to subvert conventional expectations of narrative and what film should be, by exploring non-narrative forms and alternatives to traditional filmmaking. They tend to scrutinize observations and experiences that can't be portrayed by the formal structures of mainstream cinema. These films refuse to conform to prevailing filmmaking practices and challenges the ideas about what films can show and how they can be shown. Experimental film is a genre that can't be easily described in terms of formal characteristics, therefore, interpreting these films using conventional film-theoretical framework doesn't usually work.(Peterson, 1994).

Because there are no set of rules and very often no narrative it is difficult to classify experimental films into genres that of conventional cinema. We will categorize experimental films as described in the book "Dreams of Chaos, Visions of Order: Understanding the American Avant-Garde Cinema" by James Peterson in which he describes three strains in which experimental films can be placed - poetic, structural and assemblage. Each of these strains has subgenres that either explains how these movies can be perceived or how they were created.

3.1.1 Poetic

With a set of principles of implicit viewing procedures and film construction on those of modern poetry, this experimental film strain is not so much a specific kind of film as it is an approach to the cinema. To understand the film a viewer must work out the details. At the local level, details must be integrated into meaningful units. Followed by the global level, in which viewers must integrate these meaningful units into large-scale schematic structures. To optimize the coherence viewers match the details of the film to highly structured schemata; if it fails, they try to match it to progressively more open-ended schemata. In absence of narrative, the viewers can establish some degree of overall coherence by mapping musical analogies onto the structure of the film. If viewers are unable to match even quasi-musical schemata they can still make a general appraisal of the films overall mood or atmosphere. Generally, the experimental film tends to be more open-ended than structured. (Peterson, 1994).

Coherence between shots is one of the most useful strategies for establishing local coherence in any kind of film. However, in experimental films, it is more challenging because following shots are usually shot from unconventional angles or altered by a wide range of techniques that further challenges coherence between shots in viewers minds. Most metaphors in mainstream films are diegetic, making them less

disruptive. Where is in experimental films intrusions of non-diegetic metaphors are encouraged, because experimental films typically do not create a consistent fictional world. (Peterson, 1994).

For people who are viewing poetic strain film for the first time many of the elements might be unintelligible. Therefore, when viewing such a film it is important to make the assumption that sounds and images in it have as their source the individual artist who has almost complete control over the film. In other words - personifying the filmmaker to make sense of the film. To this approach Peterson(1994) introduces following heuristics, that further help to understand these films:

- Character-as-filmmaker - filmmaker appears in his own movie, common in diary films.
- Style-as-consciousness - cinematography represents filmmakers consciousness.

Poetic films take advantage of continuity editing system, that establishes temporal and spatial links, such as match-to-action (cut from one shot to another view matches the first shot action) and eye-line match (cut from one shot to another shows what character is seeing). In commercial films these editing systems are strong cues of temporal and spatial continuity, wherein poetic films, they could be signs of thought, memory, fantasy or other subjective experience. (Peterson, 1994).

These films are organized around colours, shapes, sizes and movements in the images. Although these film can seem random, they are often organized through theme and variation. It usually starts by introducing the base relationships this film will explore, followed by segments of similar yet different kinds of relationship. This is done to build a greater distance between these segments, thus increasing the contrast as time goes on. According to Peterson(1994) poetic films can be further divided in following categories:

- Abstract
- Prototypically lyrical
- Experimental narratives

These categories can be correlated to Author-Audience Distance, scale with abstract on one side and didascalic on the other side. In this scale, abstract refers to content having the only intrinsic form with little or no attempt at pictorial, figurative or explicit representation. Didascalic, on the other hand, refers to content with high intrinsic pictorial, descriptive, explicit, figurative or narrative representational power. (Bruni, 2013).

3.1.2 Structural

The implicit viewing procedures that had evolved for the previously mentioned poetic strain do not work for the structural film strain. Sitney(2002) defined following four typical characteristics of the structural film:

- Fixed camera position.

- Flicker effect - a succession of contrasting images being quickly displayed one after another.
- Loop printing - a technique in which film is being reprinted to itself creating a continuous loop.
- Rephotography - repeated photography of the same site, with a time lag between the two images.

Nevertheless, these characteristics does not define structural film strain completely, as there are films within this strain without these characteristics. However, it does help to distinguish and generalize what structural film strain might look like.

To further understand structural film strain Peterson(1994) offers following three schemas:

- Phenomenological schema - a film is interpreted as the embodiment of some fundamental feature of consciousness. For, example a long take without appreciable dramatic action can be read as the inscription of time on the image.
- Art-process schema - film demonstrates the rigidity of conventional process of filmmaking. The viewer experiences film only provisionally; he tries to reconstruct the steps that filmmakers took to produce this film and these steps are the point of the film.
- Anti-illusion schema - film image with limited depth cues is interpreted as an assertion of the inherent qualities of the film medium. Any element of the screen that does not produce an impression of three-dimensional space is read as a demonstration the inherent flatness of the cinematic image.

Like minimalist objects, the structural film doesn't represent anything, instead, it explores the characteristics specific to the medium - movement, light, grain and spotlight elements. It shows relations between the camera and the way an image is presented.

3.1.3 Assemblage

This experimental film strain maintains a tension between its incorporated elements and the new composition that compromises them. This strain is not a single, unitary aesthetic form. It is provisional, unstable combination of elements, and to a significant degree, these elements maintain their autonomy. Furthermore, dramatizing the tension between disunity and unit is implicit. Assemblage films share many of its principles of construction with the heavily edited poetic films. Firstly, at the local level of coherence between shots are juxtaposed on the basis of a wide range of associations-metaphor, implied causality and scenographic matches. Secondly, at the global level, it ranged from very open-ended to highly structured. (Peterson, 1994)

Peterson(1994) further divided assemblage strain into following categories:

- Collage animation - can be considered as craft and as such further divided into highly crafted and bricolage. Highly crafted refers to the consistency of source imagery, coherence of space and

precision of assembly. Where is in bricolage all of the previously mentioned properties are not concerns.

- Compilation - created by combining clips from different films to create a new whole which then can be structured by global schemata that range from highly structured to a very open-ended.

3.2 Recommendation systems.

Today, big corporations such as Google, Amazon and Netflix all use recommender systems which often rely on collaborative filtering by users. Recommender systems have been developed in parallel with the web. The internet has evolved from Web1.0, to the Web3.0 as we know it today, where it collects data smart and fast in order to provide users with the most adequate information (Nath K. et.al., 2014). In a survey conducted in 2013, on recommender systems, it has been proven that recommender systems can be a useful tool for addressing a portion of the information overload phenomenon from the internet (Bobadilla J., et.al. 2013). The survey addresses the most common methods designed to tackle those problems. It divides them into three generations, depending on the evolution of the internet. In the beginning of the internet age, the first generation of recommender systems used traditional websites to collect information with the use of cookies gathering content-based, memory-based and demographics data. The second generation of recommender systems was introduced with the extensive use of the social media websites. They extensively use the Web 2.0 by gathering social information, such as friends, followers, etc. The third generation of recommender systems is already being implemented, which uses the Web 3.0 through information provided by the integrated devices on the internet (Bobadilla J., et.al. 2013). This has been already implemented by companies who push the development of recommender systems, such as Netflix or Youtube, who collect what they call implicit activity data, such as time spent watching videos, or interaction with their products (Davidson, J., 2010).

There are several different types of recommender systems that vary in terms of the functionality, the knowledge used to the algorithm and how recommendations are assembled and presented to users. The literature distinguishes between six different classes of approaches that have become a classical way to represent them (Ricci F., et.al. 2011). The main ones include Collaborative Filtering, Content-based, Demographic Based, Knowledge based, Community Based and Hybrid recommender systems. In recent years big corporations implement these approaches together, where the components of the systems are brought together overcoming known problems that occur with their development.

3.2.1 The recommendation procedure

To give further insight of how recommendations are generated the recommendation task needs to be defined. The general explanation on the task is with the following notation. If a system denotes a set of users U and a set of items I, R represents the set of ratings recorded and S represents the possible values for ratings (S = [1,5], or S = [Like, dislike]). When ratings are available, this task is most often defined as a regression, or a classification problem where the goal is to learn a function $f : U \times I \rightarrow S$ that predicts

the rating $f(u, i)$ of a user u for a new item i (Desrosiers & Karypis, 2011). This function is then used to recommend to the active user an item for which the estimated rating has the highest value.

Recommendation Equation

$$i^* = arg\ max\, f(u_a, j) \quad j \varepsilon I / I_u$$

Where i^* is the item with the highest value, u_a represents the active user. The recommendation method needs to be evaluated for its accuracy, where the ratings R are divided into a training set to learn f and a test set used to evaluate the prediction accuracy. This turns the problem into optimising problem, where the prediction for a given user is rated itself. The most common way for doing so is measuring the Root Mean Square Error (RMSE) where the lower the error, the better the performance. To achieve this, the Singular Value Decomposition technique is necessary to decrease the dimension of the feature matrix by extracting its latent factors, where latent factors can refer to the genre of the film (Koren & Bell, 2015).

MAE Equation

$$MAE(f) = \frac{1}{|R_{test}|} \Sigma \left| f(u, i) - r_{ui} \right|$$

RMSE Equation

$$RMSE(f) = \sqrt{\frac{1}{|R_{test}|} \Sigma \left(f(u, i) - r_{ui} \right)^2}$$

Methods based on nearest-neighbours are the most popular ones, due to their simplicity, efficiency and their ability to produce accurate and personalized recommendations (Desrosiers & Karypis, 2011). The most significant traditional methods, techniques and algorithms for a recommendation process in a system, as well as their relationships and groupings, can be represented as a graph.

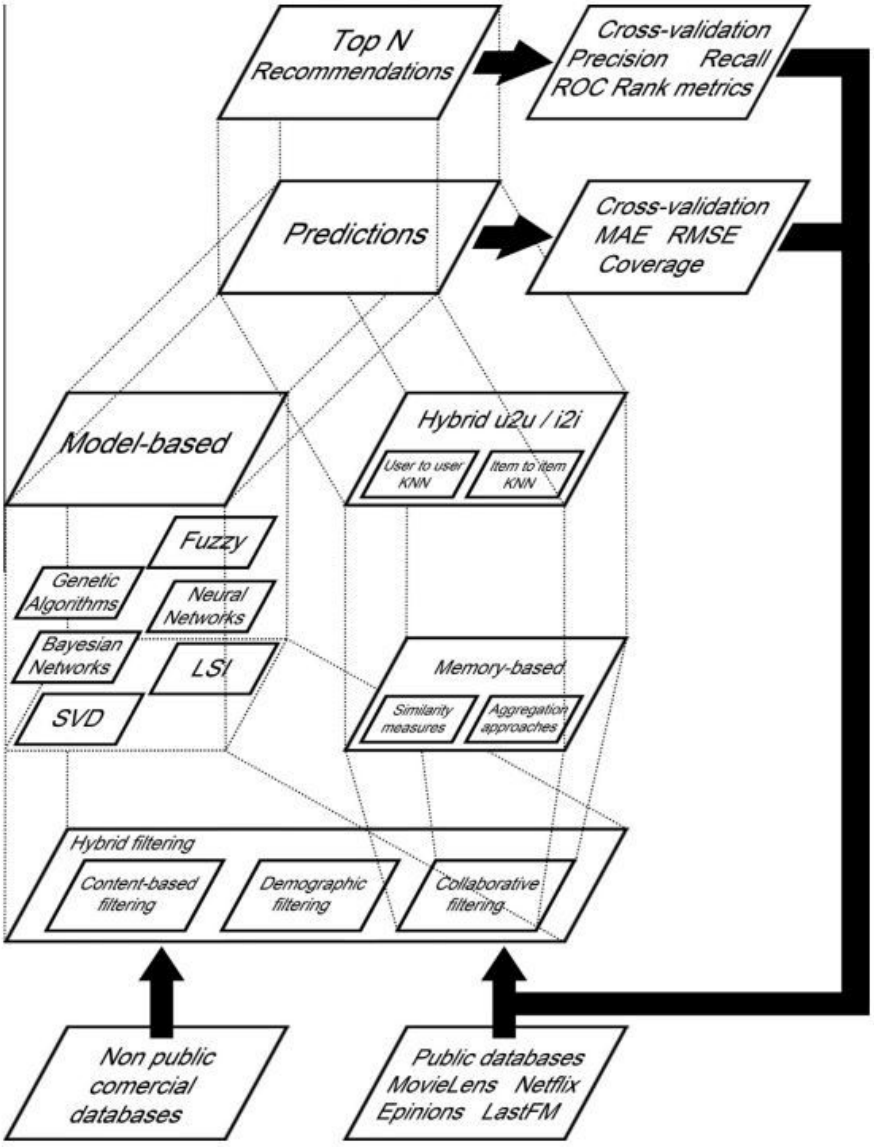

Figure 1. Traditional models of recommendations and their relationships (Bobadilla J., et.al. 2013)

As the graph suggests, some of the traditional methods (content-based and collaborative filtering) can be applied to databases, where model-based technologies make use of this kind of information. The other approach is memory-based, where, user to user; item to item, or hybrid developed systems can be used. The main purpose of both approaches is to get the most accurate prediction delivered to the user. The accuracy of these predictions may be evaluated through classical information retrieval measures, for example, precision, recall or RMSE to name a few. Cultivating these measures improves the quality of the Recommendation systems (Bobadilla J., et.al. 2013).

3.3 Collaborative filtering

Collaborative (or social) filtering approaches rely on the ratings of a user, as well as those of other users in the system. The idea is that the rating of a user for an item is likely to be similar to that of another user, if both of them have rated other items in a similar way. The goal is to predict if a user is likely to rate two items in a similar fashion if other users have given similar ratings to them (Desrosiers & Karypis, 2011). This approach is based on quality of items as evaluated by the users instead of relying on content that may be misrepresented. Systems with this approach can recommend items with very different content, as long as other users have already shown interest. The most basic example, a person who wants to see a movie, might ask for recommendations from friends. The recommendations of some friends who have similar interests are trusted more than the recommendations from others. This is achieved by taking into consideration of the friends ecosystem and applying the so-called neighbourhood-based technique. In this example several users are selected based on their similarity to the active user. A prediction for that user is made by calculating a weighted average of the ratings of the selected users (Linden g., et.al. 2003).

The main focus of this type of system is that similar users share the same interest and that similar items are liked by a user. It is a mean of recommendation based on users and their past behaviour. There are two main categories associated with collaborative filtering. User-based, which measures the similarity between target users and other users, and Item-based, which measures the similarity between the items that target users rate with other items. Currently, the proposed approaches to collaborative filtering are incorporating social information, or basic interaction with applications, using implicit, local and personal information within their ecosystem. However the main goal is to connect the personalised information from the internet of things (Bobadilla J., et.al. 2013).

The Collaborative filtering approaches can also be grouped in neighbourhood-based (memory based or heuristic based) and model-based. Neighbourhood based systems rely on the user-item ratings stored in the system are directly used to predict ratings for new items. Whereas model-based approaches use the ratings to learn a predictive model. The general idea with model-based is to model the user-item interactions with factors presenting latent characteristics of the users such as preferences (Desrosiers & Karypis, 2011). In most cases these items are the ratings themselves. There are numerous model-based recommenders, which include Bayesian clustering, Latent Semantic Analysis, Support Vector Machines or Singular Value Decomposition, to name a few.

3.3.1 Collaborative Filtering techniques.

The most widely used algorithm for collaborative filtering is the k Nearest Neighbours (kNN). The classification of kNN finds the k closest points from the training records. It then assigns the class label according to the class labels of its nearest neighbours. The underlying idea is that if a record falls in a particular neighbourhood where a class label is predominant it is because the record is likely to belong to that same class (Adomavicius G., et.al., 2015). In a recommender system the classifier, first determines k users neighbours, for the active user, then implements an aggregation approach with the ratings for the neighbours in items not rated by the initial user and extracts the predictions from the neighbours with the top N recommendations (Bobadilla J., et.al. 2013). If we want to know the class l of a point q and a

training set $X = \{\{x_1, l_1\} \ldots \{x_n\}\}$, where x_j is the j-th element and l_j is its class label, the k-nearest neighbours will find a subset $Y = \{\{y_1, l_1\} \ldots \{y_n\}\}$, such that $Y \, \varepsilon \, X$ and $\Sigma_1^k = d(q, y_k)$ is minimal. Y contains the k points in X which are closest to the point q. Then the class label of q is $l = f(\{l_1 \ldots l_k\})$ (Adomavicius G., et.al., 2015). The following is a visual representation of the kNN classifier.

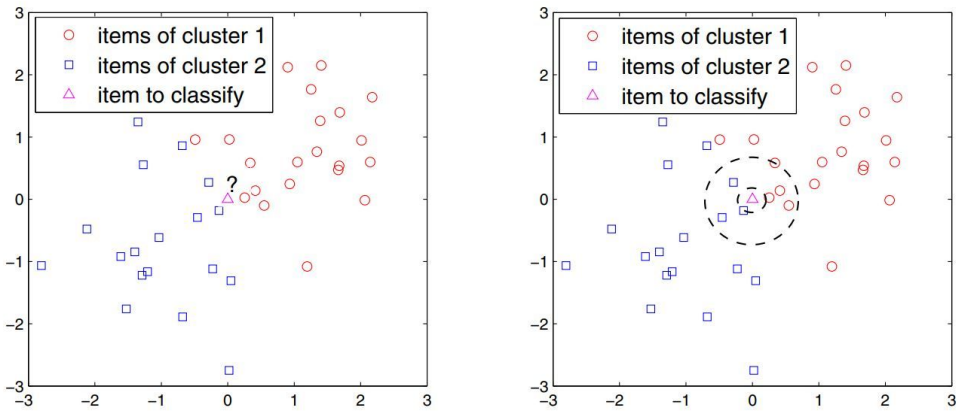

Figure 2: kNN example

The left subfigure shows the training points with two class labels (circles and squares) and the point that needs to be classified (as a triangle), noting that the point is on the boundary between the two clusters. The right subfigure illustrates the closest neighbourhood for k=1 and k=7. The point would be classified as a square for k=1, since the closest classified point is a square, and as a circle for k=5 according to simple majority vote rule (Adomavicius G., et.al. 2015).

Classification methods such as this one, and most classifier and clustering techniques, are highly dependent on defining an appropriate similarity measure or distance measure. Such measures include Euclidean distance, or the Mahalanobis distance. In recommender systems, Pearson correlation coefficient and Cosine similarity are most commonly used, since most systems consider items as document vectors of n-dimensional space and compute their similarity. (Adomavicius G., et.al. 2015). In short, the similarity between items can also be given by their correlation, which measures the linear relationship between objects. The following equations explain these similarity measures:

Pearson Correlation:

$$Pearson\,(x,y) = \frac{\Sigma(x,y)}{\sigma x * \sigma y}$$

Given the covariance of data points x and y Σ and their standard deviation σ.

Cosine similarity:

$$cos(x,y) = \frac{(x \bullet y)}{||x||||y||}$$

Where $\bullet$ indicates the dot product and $||x||$ is the norm of vector x.

In a system this computes the similarity between users and user based Collaborative Filtering. If we consider u_{i, k} to be the similarity between user i and k, and v_{i, j} denotes the rating that user i gives to item j with v_{i, j} = ? if the user has not rated that item. The main difference is that Pearson correlation, as compared to Cosine similarity is invariant to adding a constant to all elements. In this case the methods can be expressed as:

Pearson correlation:

$$u_{jk} = \frac{\sum_{j} (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\Sigma j\ (v_{ij} - v_i)^2 \Sigma j (v_{kj} - v_k)^2}}$$

(Introduction to Recommender systems, 2014)

Cosine similarity:

$$cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v^2_{ik} \sum_{k=1}^{m} v^2_{jk}}}$$

(Introduction to Recommender systems, 2014)

In general, collaborative filtering approaches are based on user demographics or preferences, but the traditional approaches often bump into issues, such as the cold-start problem or the sparsity problem (Billsus & Pazzani, 1998) Studies on this topic have shown that users interacting with recommendation systems need to provide initial information, or preferences, before interacting with the system (Ricci F., et.al., 2011). This implicit information can be extracted with an initial interaction, or gathered through social media for example. But even in this case, the system still experiences difficulty of accurately predicting the recommendation, as it is dependent on the interaction of the user. On the other hand, as some websites provide a very broad variety of items, the recommender system may not suggest items which have not yet been rated. The system itself may be experiencing difficulty, by the sheer number of items that are inside its own database (Linden g., et.al. 2003). So some items may never even be

recommended even though they can be a perfect suggestion to some users, just because of the fact that they were not rated.

- **Cold-start Problem**

The cold-start problem occurs when it is not possible to make reliable recommendations due to an initial lack of ratings. As mentioned above there are two common categories associated with collaborative filtering. Since new users in the system have not rated any item they cannot receive any personalised recommendations. Thus the system has to wait until all users rate enough of items before making an accurate recommendation (Bobadilla J., et.al. 2013). On the other hand, new items entering the system will also not be rated by the users, so they go unnoticed by a large part of the ecosystem. This enters a cyclical flaw where users are unaware of the items they do not rate and the items themselves are left out never to be rated.

To overcome this problem some systems prefer to ask their users for their favorite movies and rate them, before letting them use the application (Ricci F., et.al., 2011). This is the easiest way to overcome the User Cold-start problem. The item Cold-start problem can be easier to manage, as every item has a predefined number of categories, which are set up beforehand. This rationalises the items and the system will be capable of suggesting items with certain aspects to the users, without the need to rate them previously (Linden g., et.al. 2003).

- **Sparsity Problem**

This problem occurs when the amount of items become very large, reducing the number of items users have rated. This creates a very large gap where users may never find items that they may actually rate positive. In this case it is highly unlikely two people may have common interests, making the correlation coefficient less reliable. In other words, in some recommender systems the existing number of items exceeds the amount a person is able to explore. This makes it hard to find items that are rated by enough people on which to base predictions. If a recommendation of an item is made using neighbours computed from a small amount of ratings, then the accuracy will be lower (Billsus & Pazzani, 1998).

The most frequently used strategy to tackle these problems consists of turning to additional information. This additional information is usually content based where an item may have a description or tags that define it. This way it is possible to make recommendations based on the data available for each user. Overcoming these problems may be done through Implicit ratings, dimensionality reduction, or using the description of the products. This creates another type of algorithms that can be classified on their own. These strategies are often using hybrid systems, that incorporate for example content based, demographic based, or social based approaches, to name a few (Lops P, et.al. 2011).

- **Matrix Factorisation**

As mentioned, the sparsity problem created by the Collaborative filtering approach is of a great concern. To reduce the high levels of sparsity in Recommender systems some studies suggest dimensionality reduction techniques based on model-based approaches (Bobadilla J., et.al. 2013). These techniques are very capable for processing large databases and providing scalable approaches (Billsus & Pazzani, 1998).

For example the Principle Component Analysis (PCA) is a classical statistical method to find patterns in high dimensional data. This technique can reduce the dimensionality of the data by neglecting instances with a small contribution to the variance of the features. Earliest Recommender systems have utilised PCA and has been improved over time, but current trends seem to indicate that other matrix factorizations techniques, such as Singular Value Decomposition (SVD) are preferred (Amatriain X., et.al., 2015). Recently matrix factorisation models have gained popularity, thanks to their attractive accuracy and scalability and SVD techniques provide good prediction results (Koren & Bell, 2015).

In information retrieval SVD, is well established for identifying latent semantic factors, as the technique serves as the guiding stone for such analysis . Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterising both products and users on factors automatically inferred from user feedback. In the case of this project, where the products are movies, factors might measure obvious dimensions such as comedy vs.drama, amount of action, or orientation to children (depending on the description of the movie), or sometimes less well defined dimension like character development, or completely uninterpretable dimensions, such as abstract or avant-garde films (Koren & Bell, 2015, Amatriain X., et.al., 2015).

SVD is represented with the following theorem: It is always possible to decompose a given $n \times m$ matrix A (n items, m features) into $A = U\lambda V^T$. An item x features matrix can be decomposed into three different ones: an item x concepts, a concept strength, and a concept x features. This can be then be represented into an $n \times r$ matrix U (n items, r concepts), an $r \times r$ diagonal matrix λ (strength of each concept), and an $m \times r$ matrix V (m features, r concepts). To illustrate this idea the diagonal matrix λ contains the singular values, which will always be positive and sorted in decreasing order. The U matrix is interpreted as the "item-to-concept" similarity matrix, while V matrix is the "term-to-concept" similarity matrix (Amatriain X., et.al., 2015).

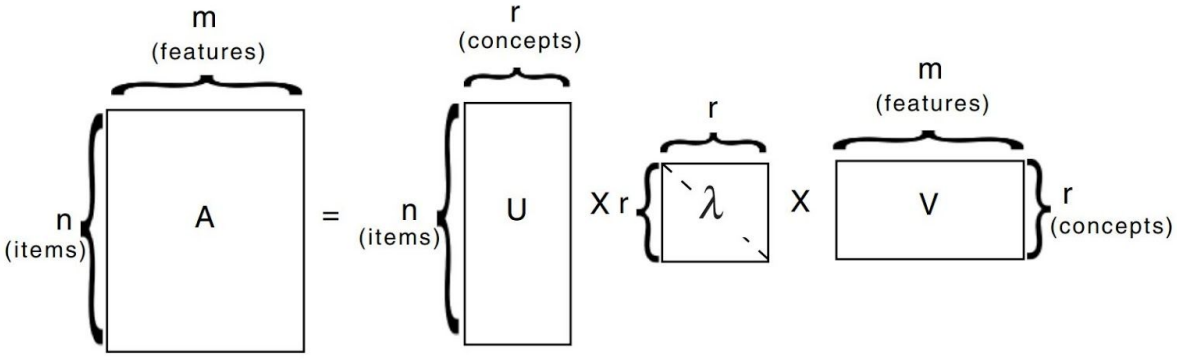

Figure 3: Illustrating the basic SVD Theorem.

3.4 Content-based filtering

Content-based (or cognitive) systems try to recommend information similar to the ones a given user has liked in the past, based on his/her preferences. In a nutshell, the system matches the attributes of a particular user's preferences and interests, with the attributes of a content object, in order to recommend the user to new potentially interesting content, that the user might not have otherwise discovered (Lops P, et.al. 2011). Such a system recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors, or terms, typically additional information provided to describe the item (Adomavicius G., et.al. 2015). Items that are recommended to the user are represented by a set of features. In the case where the recommender system suggests movies, the features that describe a movie may be actors, directors, genres and so on. Often in such filtering systems item descriptions are textual features extracted from the internet of things. The textual features create a number of complications, due to natural language ambiguity (Lops P, et.al. 2011). Systems implementing a content based approach analyse those descriptions of items previously rated by a user, and build a model or profile of user interests based on the features of the objects rated by that user.

The common principle of content-based approach is to identify the common characteristics of items that have received a favorable rating from a user and then recommend to him new items that share these characteristics. Using the content of an item instead of the item itself could increase the amount of information people have in common. As user-based collaborative filtering is very simple, it doesn't take into consideration that users` preferences can change over time, indicating that the calculated neighbours in the kNN algorithm, may lead to bad performance (Amatriain X., et.al., 2015). With the addition of more information, such as descriptions the recommender system uses Content based filtering. It successfully avoids the problem posed by dynamic user preferences as the descriptions of items are static. Using such systems with a categorical correlation of contents, helps companies provide the most valuable information to their customers. For instance, in the movie industry films are categorised by experts and directors into genres (content-based groups) in order to be more structured in meeting the desires of the consumers (Lops P, et.al. 2011).

Items with information describing their nature take this information into feature vectors. The vector which represents items with a description in the form of text documents, such as news articles or web documents often contains The Term Frequency Inverse document frequency weights of most informative keywords (Desrosiers & Karypis, 2011).

- **Keyword-based Vector Space Model**

A very simple way to create a recommender system with content based filtering, is with the use of simple retrieval models, such as keyword matching or the Vector Space Model with the basic Term Frequency - Inverse document Frequency weighting (TF/IDF), which extracts textual information from features (Lops P, et.al. 2011). The Vector space model alone can raise issues related to the weighting of the terms and measuring the feature vectors, TF-IDF is used as a weighing scheme, which is based on empirical observations regarding text. In general, this algorithm looks at which terms are more likely to be relevant

to the topic. This can be achieved if terms occur frequently in one instance, but rarely in the rest of them. This can be expressed in the following function:

$$TF - IDF(t_k, d_j) = TF(t_k, d_j) \cdot log\frac{N}{n_k}$$

In this function N denotes the number of documents in the data, and n_k denotes the number of documents in the collection in which the term t_k occurs at least once. Then, $\mathrm{TF}(t_k, d_j)$ computes the maximum over the frequencies $f_{z,,j}$ of all terms t_z that occur in document d_j :

$$TF(t_k, d_j) = \frac{f_{z,j}}{max_z f_{z,,j}}$$

In order for the weights to fall in the [0,1] interval and for the documents to be represented by vectors of equal length, weights obtained by the TF-IDF equation are usually normalized by cosine normalization like the following which enforces the normalization assumption:

$$w_{k,j} = \frac{TF{-}IDF(t_k,d_j)}{\sqrt{\sum_{s=1}^{|T|} TF{-}IDF(t_s,d_j)^2}}$$

As content based usually depends on information like description keywords or tags, that are often stored as text, they can be morphed to represent feature vectors, which then can be measured by similarity scores. This measure must be computed to determine the closeness between the feature vectors. Many similarity measures have been derived to describe the proximity of two vectors. In the content-based recommender systems relying on VSM, both user profiles and items are represented as weighted term vectors. Predictions of a user`s interest in a particular item can be derived by computing the cosine similarity measure, which is the most common way to do this (Lops P, et.al. 2011).

Such keyword based recommender systems have been developed in a relatively short time for various fields of applications. Each application presents different problems that require different solutions (Lops P, et.al. 2011). In the case of movie recommender systems, the use of text categorisation techniques teaches movie descriptions, obtained from the Internet Movie Database (IMDB) or the Movie Database (tMDB), suggesting movies relevant to the users interests. In order to get recommendations, the user is initially asked to rate a minimum number of movies. This way the system integrates voting schemes, designed to allow multiple individuals with conflicting preferences to arrive at an acceptable compromise and adapt them to manage conflicting preferences in a single user. In other words compiling both the user`s ratings and the keyword overview ensures more accurate recommendations by the system, boosting the performance of the system.

Improving the Recommender system

Combining both Collaborative filtering and Cognitive filtering could take advantage from both the representation of the content as well as similarities among the users. Most hybrid approaches combine the

two types of information, but sometimes, but using the two filtering techniques independently is also possible. This innovative approach also ensures the overcoming of the cognitive problems, where the system has difficulty in distinguishing between subjective information such as points of views and humor (Lops P, et.al. 2011).

4 Current trends in the market

4.1 Netflix

When Netflix was still a DVD renting company, its recommendation system was based on predicting how many stars their customers would give to a movie. In 2006, Netflix tried to improve their algorithm by launching the “Netflix Prize”, a data mining and machine learning competition that would optimize and improve their five-star rating recommendation system. After two years, a team won the price by creating an algorithm that was 10% better than the current. However, Netflix never incorporated the vining algorithm because of two reasons. Firstly, they changed their business model and transferred from a DVD rental company to an online video streaming service, which generated greatly larger data and would require much more complicated algorithms. Secondly, and most importantly, they realised that they were trying to answer the wrong question by predicting how many stars would a customer give to a movie. The real question they wanted to answer now was - which movies should be recommended to the specific user, in order to make them use their service for a longer time and continue being subscribed. (Zhou, 2008).

By transition into a digital platform, Netflix gained access to vastly larger customer data both explicit and implicit. Implicit data that Netflix gathers from customers consists of: search queries, movie views, ratings, shares, downloads and those movies selected to been watched later. The explicit data provides Netflix with more complex data such as: how customers use the service, their scrolling behaviour, do they watch the whole movie or stop after first few minutes. Furthermore, data of where, what time and from what device movie is being watched also feeds the recommendation system to predict with even higher accuracy movies that their customers would like to watch.

Every movie on Netflix have complex set of metadata attached to it. That includes information such as title, genre, synopsis, cast, links to thumbnail images, trailers, and subtitles are linked to each of the movies. Furthermore, Netflix have a dedicated team with people that tags movies with hundreds of keywords that describe the movie based to specific guidelines. Currently, there are around 100,000 sub-genres generated by compiling key-words, genres and cast, providing unique ways to describe type of movies. Netflix does this to further personalize each customers home page with content that they will like based on their viewing habits.

Netflix also uses customer data to predict content acquisition. With Netflix being available in over 190 countries, this have become highly important for the company. Before Netflix can add a movie to their catalogue, they need to obtain a license that can be region or country specific and are often held to terms for years at a time. Ultimately, giving customers around the world the same content would be great, but

cost for it would be to large. Predicting which movies to licence to which countries, allows Netflix to invest smart and provide customers around the world with content they like.

Being able to predict success of new movie even before its production is a game changer in movie business, because it allows to invest large amount of money to create a new content with minimal risk. For example, Netflix saw a pattern that many customers were interested in political movies and actor Kevin Spacey, which gave rise to one of the most watched Tv-series - House of Cards. Furthermore, in the beginning they targeted marketing for this Tv-series towards those exact customers, ensuring that show gains momentum and positive feedback. In addition, Netflix has set goal to release new content every month that appeals to their customers and is on a mission to have half of its content to be original.

A problem that Netflix recommendation system is facing, is how to make sure that customer will find what they are looking for and understand why they chose that particular movie. Consumer research suggests that Netflix customers lose interest after 60 to 90 seconds of choosing, having reviewed 10 to 20 movies or tv-shows. The customer either finds what he was looking for or abandons the service. Netflix user interface contains multiple rows with predicted themes that might be interesting for a particular customer. Depending on a device that Netflix is used on, there are usually around 20 to 40 rows on each page and up to 75 videos per row. These numbers vary because of user experience considerations. The videos chosen for each row represents the best recommendation for each customer. However, customers have different preferences from session to session, and many accounts are shared by multiple customers. By providing a diverse selection of rows, Netflix makes it easier for the customer to find what they were looking for. Netflix recommendation system is used on the majority of pages, beyond the home page and influences about 80% of hours spent watching videos. The remaining 20% comes from the search query, which requires its own set of algorithms. (Gomez-Uribe, 2016).

Netflix uses a Top N video ranker that produces the recommendation in the Top Picks row. The goal of this algorithm is to find the best-personalized recommendation in the entire catalogue for each customer. There is two type of trends that this ranker can identify very good: events that repeat yearly, such as romantic movies for Valentine's day and short-term events such as natural disasters in populated areas, which result in increasing interest in documentary movies for such an event. (Gomez-Uribe, 2016).

Evidence selection algorithms evaluate all the evidence items, that can be shown for every recommendation, to select few that might be most helpful to the customer. For example, it decides whether to show that a movie has won an Oscar or instead show the customer similar movie to a recently watched. This algorithm also decides which thumbnail image out of several versions use to better support a given recommendation. (Gomez-Uribe, 2016).

Another type of categorization is Because You Watched (BYW), which shows recommendations based on a single video watched by a customer. The video-video similarity algorithm is un-personalized and computes metadata and popularity similarity for watched video. Although video selection is not personalized, the choice of which BYW rows make it onto a homepage is personalized, and the subset of BYW videos recommended in a given BYW row benefits from personalization, depending on what subsets of the similar videos are estimated that the customer would enjoy. (Gomez-Uribe, 2016).

Sub-genre rows such as Exciting Movies shown in the Figure xxx are driven by Personalized Video Ranker (PVR) algorithm. This algorithm orders the entire video catalogue or subsets selected by filtering for each customer profile depending on their preferences. This is why videos in same genre row will be different for each user. PVR works better when the personalized recommendation is blended with un-personalized popularity, which is used to drive popularity row.

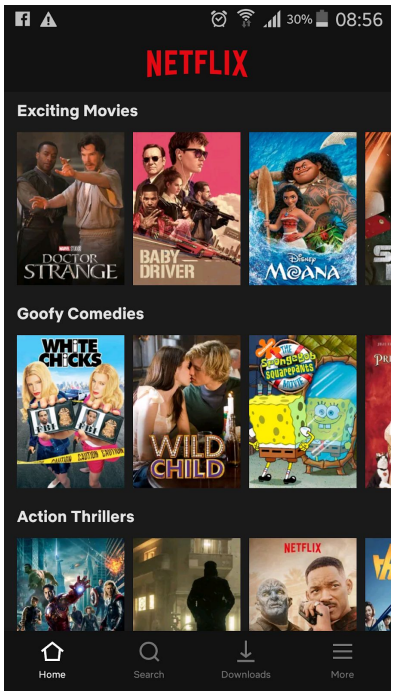

Figure 4: Netflix Home Page

To measure customer retention rates Netflix uses A/B tests, to compare medium-term engagement together with customer cancellation rates across algorithm variants. Algorithms that improve these A/B test metrics are considered better. Firstly, quick iterations to prototype an algorithm are done through offline tests by analysing historical data to quantify how well a new algorithm can predict previous positive customer engagement. The assumption is that customers would have the same behaviour when interacting with Netflix. Once, offline experiments show decent results, the new algorithm is put to an online A/B test that randomly assigns different customers to different algorithms, Netflix refers to- these as cells. Each cell in the test is mapped to a different PVR algorithm, one of which refers to the default algorithm to serve as the control cell in the test - the others cells are the test cells. These tests are conducted over a period of 2 to 6 months in which customers interact with the system. Because subscription model allows subscribing for a minimum of one month it is necessary to conduct this test for at least two months to see cancellation rates. The resulting data is analysed trying to answer following questions:

- Are customers finding the part of the product that was changed relative to the control more useful? For example, are they finding more videos to watch from the PVR algorithm than in the control?
- Are members in a test cell streaming more on Netflix than in the control? For example, is the median or other percentile of hours streamed per member for the duration of the test higher in a test cell than in the control?
- Are members in a test cell retaining their Netflix subscription more than members in the control?

These tests are an important source of information for making product decisions. Usually these tests are very informative, however, despite the statistical sophistication that goes into their design and analysis, interpreting A/B tests remains partly art. (Gomez-Uribe, 2016).

4.2 YouTube

With over billion users creating, sharing and discovering video content YouTube presents one of the largest scale and most sophisticated recommendation systems in the world. Users come to YouTube for many reasons which span from less to more specific: To watch a video that they found in other websites, to find a specific video around a topic of their interest, or just to be entertained by content that they find interesting. Personalized video recommendation system is one of the ways to address this last reason. (Davidson, 2010), (Covington, 2016).

In their recommendation system, a number of data sources are considered. Firstly, content data such as the video streams and metadata such as title, description and many more. Secondly, user activity data, which can be divided into implicit and explicit categories. Implicit data is the result of users interacting with the service and watching videos, e.g. how much of the video user watched. Explicit data is the result of user activities such as liking a video or subscribing to its uploader. Instead of selecting only most relevant videos YouTube creates a balance between relevancy and diversity across different categories. Because the majority of users have interest in different topics at different times, videos that are too similar to each other are removed at this stage to increase diversity. They achieve this by imposing constraints on the number of recommendations that are too similar to a video, or by limiting the number of recommendations from the same uploader. It is important to consider how these videos are shown to the user. Firstly, recommended videos are displayed with a thumbnail image and their title, together with information about the video, its age and popularity. This helps users to decide quickly whether they are interested in a video or not. Furthermore, YouTube provides the user with an explanation and link to the video that gave the recommendation. (Davidson, 2010), (Covington, 2016).

Similar to Netflix, YouTube uses online A/B tests as the main method for evaluating recommendation system performance. In this test, users are divided into distinct groups where one group is set as control baseline and other groups are then compared against each other over a set of predefined metrics and even swapped for another period of time to eliminate other factors. Since evaluation takes place in the context of the actual user interface, it shows user actual behaviour, which is a big advantage compared to test which would be set in a controlled environment. There is an issue when comparing the recommendation

system performance to other modules that results in presentation bias in which recommendations are placed at the top by default. To solve this YouTube looks ar metrics from the browse pages and compares recommendations to other algorithmically generated video sets: videos that have received most numbers of views in a day, videos that users have added to their collection of favourites and videos receiving most like ratings in a day. (Davidson, 2010), (Covington, 2016).

5 User Interface Design

The user interface design is centred around assumptions of what users might need and want to do, which in our case is the ability to easily find interesting films, gain more information about them and be provided with personalized suggestions. In order to satisfy user needs the interface should have elements that are easy to access, understand, and use. Nowadays, users have become familiar with many interface elements functioning in a certain way, so it is important to create a function which is consistent and predictable. Since Netflix is the largest video streaming service it was used as the base to create Re:voir user interface.

When designing Re:voir user interface the relationships between elements and their layout were considered to draw the user attention towards the most important pieces of information. The colour was selected to be blue because it is currently used by Re:voir, creating consistency in the brand. Reason Choosing only dark monochromatic tones of blue throughout of design was to create less strain on user's eyes when using this application for long periods of time and further not to overwhelm users with colours since film thumbnail images are colourful by themselves. Each element in the interface has a specific colour assigned to it, which creates consistency and directs user attention. The typography is also considered with four different fonts from the same family and four different font sizes, that similar to colours, guides users through the interface and in combination creates a pleasing visual design.

Figure 5: Main Page

5.1 Functionality

Every page in this application has a header that is pinned to the top on every page so that users could have immediate access to main navigation, without the need to scroll back to the top. In the main page(Figure xxx) header contains menu button, company logo and search button. The menu button on the top right corner opens a menu page through which users can change the profile and their personal settings, view their film list and history of films they have already seen. Once in menu page, this button is changed to a close button, that brings the user back to the homepage. The Re:voir logo in the middle has two functionalities - it promotes brand recognition and also serve as a button which brings users back to the main page from any other page in the application. The search button opens a page in which users can search for their desired content through the search query, titles, genres, years and keywords. This button is only accessible on the main page. In other pages, it is substituted by either share button or filter button. Share button appears on film and author pages allowing users to share selected page in social media. Filter button appears on pages that contain lists, such as titles, authors, keywords, genre and subgenre of selected films.

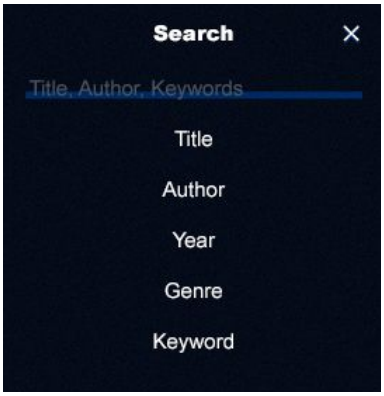

Figure 6: Search Page

The search page (figure xxx) contains:

- The search query in which search engine tries to match the letter combinations to film titles, author names and keywords, showing a list of most relevant results. Selecting a result opens corresponding page either film, author or list of films containing the keyword.
- Button to the title page in which a list of all film titles is shown in alphabetical order, that can be further filtered. Selecting a film title opens the corresponding page.
- Button to author page in which a list of all authors appears in alphabetical order of their last names, that can be further filtered. Selecting an author opens the corresponding page.
- Button to year page in which a list of all decades appear (1940's, 1950's etc.). Selecting a decade title opens a page with films released within that decade.
- Button to genre page in which a list of all genres and subgenres appear. Selecting a genre opens a page with films within that genre.
- Button to keyword page in which a list of all keywords are shown in alphabetical order, that can be further filtered. Selecting a keyword opens a page with films containing the selected keyword.

Figure 7: Filter Page

The filter can be applied to pages with lists of films, authors and keywords. It can be accessed by the button placed in the same location as the search button. This page (figure xxx) contains:

- Recommended For You - uses the recommendation system to arrange list starting with content most relevant to the user.
- Title A-Z - arranges content alphabetically starting with letter A using the first letter of film, keyword or authors surname.
- Title A-Z - arranges content alphabetically, starting with letter A, using the first letter of film, keyword or authors surname.
- Newest first - arrange films by their release year, starting with the newest one.
- Oldest first - arranges films by their release year, starting with the oldest one.
- Popularity - uses the weighted rating system to arrange content by its popularity.

For retrieve popularity, a Weighted Rating system (WR) is used, which is described by following formula.

$$WR = \left(\frac{v}{v+m} \times R\right) + \left(\frac{m}{v+m} \times C\right)$$

Figure xxx WR formula

In which v is the number of votes for the movie

- m is the minimum votes required to be listed in the chart
- R is the average rating of the movie
- C is the mean vote across the whole catalogue

The main page (Fig xxx) contains two sections:

- An image below the header is curated by Re:voir and can be linked to any page within the application or to an external webpage. The purpose if it is to provide information about promotions and events that Re:voir wouldn't want their users to miss since it is placed in the most visible place of the whole application.
- Multiple rows of films that can be relevant to the user and change depending on their watch behaviour. These rows can be an author, genre, subgenre, year or keyword specific, or contain combinations of all of them.

For the new users, who haven't yet used the platform, therefore, not provided any information to recommendation system rows shown would not be personalized. These rows would contain:

- Popular films, that would be arranged using previously mentioned weighted average rating system of other user ratings.
- New to Re:voir - films arranged by the time they have been added to the catalogue.
- Genre rows - showing rows containing poetic, structural and assemblage films arranged by their popularity.

Once recommendation system would have enough data to make the predictions more rows will be added, each personalized towards their interests. The main row would be “Recommended For You”, which would provide users with films most relevant to them. Furthermore, there could be many different combinations of custom rows. These combinations would be constructed as following - [Keyword]+[Genre/Sub-Genre]+[Author]+Films From[Year]. For example, if a user watches a lot of films by author Jonas Mekas and also films within the poetic genre, a row “Poetic Jonas Mekas Films” would appear and show films that contain both of these elements in their metadata. If a custom row gets minimal attention from the user it will be substituted with a different personalized row tailored towards user interests.

When users select a film from the row it opens a page (Figure xxx), in which user can watch the film, share it in social media, open add to their film list, download to be viewed offline and rate. A user can also open this films author page and pages of genre/sub-genre and keywords attached to this film, e.g. poetic film list. Furthermore, additional information about the selected film is shown. such as title, author, summary, genre/sub-genre, year and runtime. Lastly, a row in the bottom would be populated by films similar to it using content-based recommendation system.

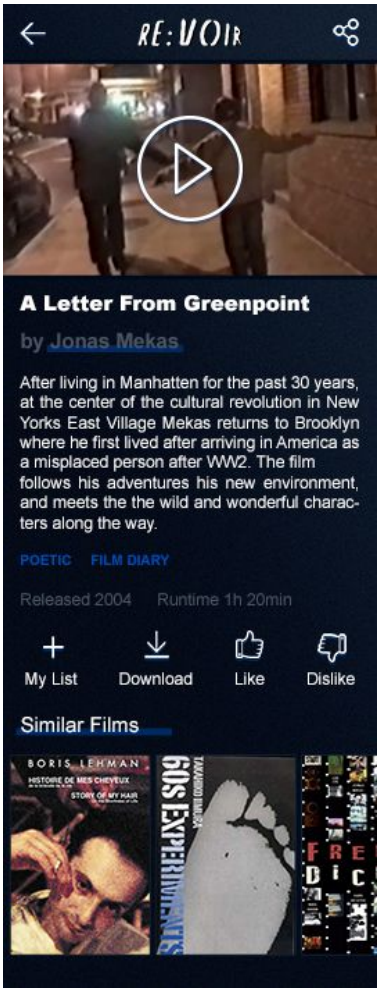

Figure 8: Film Page 1

Author page 1 (Figure xxx) can be accessed through the search query, author page, their film page or personalized row. This page contains a picture of the author, their name, short biography, row of their films and row of films similar to this author. A user can also share this page in social media. It is important to be able to access information about the specific author because understanding who there are, creates a better appreciation of their films which even more so important when viewing experimental films.

Figure 9: Author Page 1

Each of the genres, sub-genres, keywords, decades and personalized rows have a list page (Figure xxx). In this page information about each of the films are shown next to the thumbnail image, allowing the user to gain more information about the film without selecting it, which is not available in rows due to user experience issue of overcrowding the view. Furthermore, a filter button is added so that user can more easily find what they were looking for.

Figure 10: Film List Page 1

This user interface should be further improved through usability tests in which selected group of users would interact with it by completing tasks, and analysis of their behaviour and success would guide the decision to optimize it and create more user-friendly. Furthermore, responsive design guidelines should be created so that users would have coherent experience using different screen size devices.

6 Development of the Recommender System

6.1 Methodology used

Software development is the process of developing software product in a planned and structured process. It involves creating a computer program, or a set of programs to perform 2 various tasks, as well as maintaining and improving these programs. This software lifecycle (creating, maintaining, improving) is one of the most important concepts for the methodological development of computer programs (Schach R., 2002). Following a methodology while developing the product, will in turn help visualise its current state, and will serve as a strategy to move forward in the development phase. This helps predicting the outcomes, or discover underlying patterns, all to gain insights leading to actions that can improve the future outcomes of the project (Rollins 2015).

Developers must choose to follow from two main models which can help them get to the desired result of a project. The first model is the waterfall development model, where developers follow a strict schedule with requirements and deadlines, and the other is the agile development model, where developers solve issues as they arise (Schach R., 2002). As this project is considered a software application which

recommends movies, Agile development is highly recommended. Agile (as opposed to the normal Waterfall approach) incorporates the vital iterative stages, which are necessary for a project such as this one. (Cockburn, A. 2006). Agile Software development has given a push to real software development process improvement and elaboration. The main core of this model is in adapting the process of product development throughout its life cycle. Here the project can be easily adjusted to custom changes throughout its development. This is important because this method uses the iteration cycle.

The methodology used for creating the technical machine learning part of this project is going to be the *"Foundational methodology for Data Science"* proposed by IBM Analytics in 2015. This methodology sets the foundations which will serve as a guiding strategy for solving the problems that arise. This section will generally explain the methodology and its fundamental steps, which in turn will help explain the current state of the project in detail. The following diagram depicts the 10 stages of the methodology proposed, which represent an iterative process leading from solution conception to solution deployment, feedback and refinement (Rollins 2015).

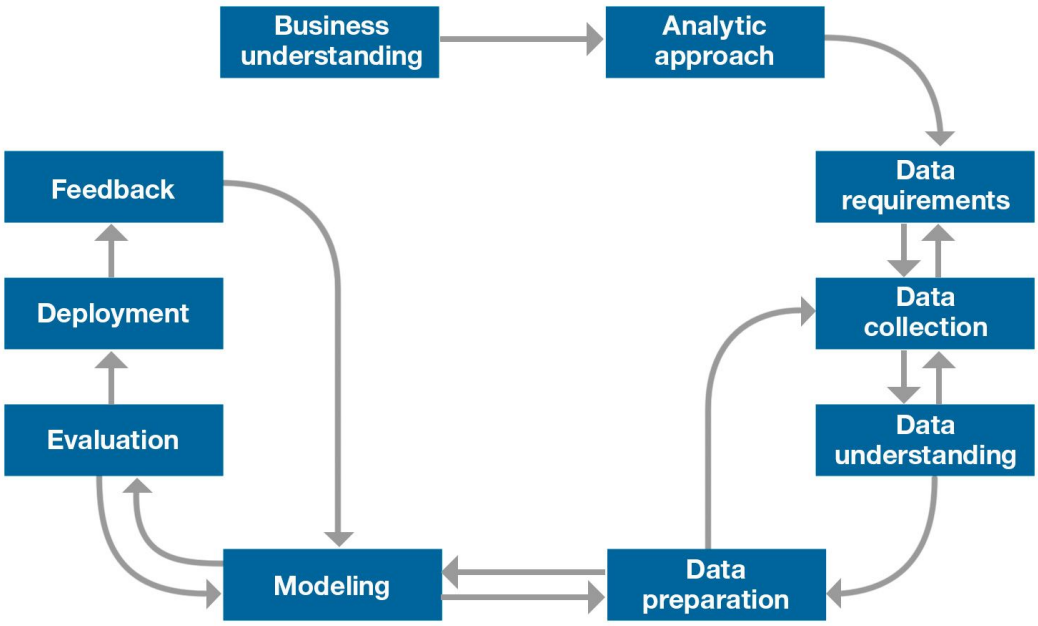

Figure 11: Data Science Methodology.

As suggested by the diagram, this methodology incorporates the business idea and suggests the iterative agile development. Both are highly necessary for this project. The 10 stages will be broken down in a more compact way, dividing them into sub groups, and grouping them into Primary phases. Primary phases are the key points to follow as progressing in the development. They are 1. Formulating a problem, 2. Data Gathering, 3. Data Analysis, 4. Production. This way, managing the project will be easier, helping to understand the further details of the problem.

- **Formulating a problem**

Business understanding

The Formulating a problem phase is the beginning of every project. In order to solve the problem, it needs a background information on the business itself. By having a clear overview of the background, the problem is easily defined, laying the foundations for achieving the desired outcome. This leads to better understanding of the project objectives and the solution requirements. In this project the background includes the Re:Voir business model, background of Experimental Films field and the understanding of the state of the art recommender systems overview.

Analytic approach

In the analytic stage, after clearly stating the problem it is easy to pinpoint the objectives necessary for the project. Here identifying the techniques for achieving the desired outcome is crucial. This includes the analysis of Experimental Films and their respective categorisation, understanding of each recommender system approach and its implementation in the project.

- **Data Gathering**

Data requirements

After the state of the art analysis has been reviewed, the next step will determine the data requirements for the analytic methods that are going to be used, which may require particular data content. In the scope of this project, the data required will be driven by the needs of the company. The data is also, going to revolve around movies which have the predetermined features necessary for the recommendation system.

Data collection

After knowing the requirements the necessary data resources can be easily gathered. Since the movie industry is large, there are all kinds of resources that can be used. The ones that will be chosen for this project also need to resemble the features that the Re:Voir database is using at the moment.

Data understanding

Understanding the data`s contents, by assessing the quality and discovering insights by the use of statistics and visualisation techniques can further close the gaps of understanding the problem which leads generally to good decision making in the analysis phase.

Data preparation

The data preparation stage comprises all activities used to construct the data set that will be used in the modeling stage. These include data cleaning combining data from multiple sources and transforming data into more useful variables.

- **Data Analysis**

Modeling

The modeling process is highly iterative. It can be considered as the development process in the project. After the dataset has been gathered and prepared, it is easy to develop a predictive or descriptive models using the analytic approach mentioned previously. In the case for this project, the modeling phase is applying the methods and techniques to create the recommender system.

Evaluation

After each model is applied it will be evaluated using the evaluation techniques respectively. This way the system will ensure to generate quality recommendations and it will be checked for weather it addresses the problem which was set up by Re:Voir appropriately.

- **Production**

For this project, this stage of production will include the testing and evaluation of the system. The system will be tested on film major students in Dongguk University Seoul (South Korea). The fact that students who are studying in the same field as the scope of this project, ensures that the results gathered provide a solid ground of criticism. This criticism is helps in the evaluation of the performance of the system, and observes how it will affect the deployment in the future. The observations, done with a help of a questionnaire, will test the capability of the project, helping the company understand how to increase the accuracy and its usefulness in the future.

6.2 Gathering data

Re:voire is a company that sells experimental movies as items, hence the data that this project will be developed with has to mimic a movie data like the information stated above. This way will help ensure that the project is developed the same way as with their database, hence the data that is going to be used has to be meaningful for the needs of this project and its goals. The data which will be used for this project comprises of two datasets. The MovieLens dataset and the tmdb 5000 movies provided by Kaggle. In order to to create the system around the company's needs, the current database format needs to be understood. After that, a bridge that connects the datasets will be defined, which will enable the system to meet the needs of the company. Re:Voir`s catalogue consists of 364 movies, each of these movies have the following information attached to them:

Title	Free Radicals
Author	Pip Chodorov
Summary	A documentary film by Pip Chodorov on the history of experimental film.
Description	Here is a film that shines with its cast: Hans Richter, Maya Deren, Stan Brakhage, Jonas Mekas, Maurice Lemaitre and other “star” experimental filmmakers (mostly americans, up through the ‘70s) parade in majesty. They do so with no false modesty (their candidness and confidence are rather surprising, sometimes humorous), and with a clear pleasure at having their turn in front of the camera.
Format	DVD9 PAL Interzone/Region-free, Stereo, 16:9, color/couleur
Year	2011
Language	English, French

Subtitles	French, English, Spanish, Serbian, Czech, Lithuanian, Japanese, Chinese, Korean
Runtime	83 min
Publisher	RE:VOIR
Keywords	Film diary, Documentary

Table 1: Re:Voir movie data

This information allows the company to divide movies into subcategories that helps users to find what they were looking for. These subcategories are:

- By filmmaker - which is further divided by in increments A to B, C to D etc. This categorization utilizes name of the author attached to each movie.
- By keyword - each movie can have none, one or couple keywords attached to them
    - Abstract
    - Animation
    - Compilation
    - Contemporary art
    - Dance
    - Documentary
    - Fiction
    - Film diary
    - Found footage
    - Landscape
    - Poetry
    - Portrait
    - Structural
    - Surrealism
- By label - utilizes publishers name attached to each of the movies.
- By period - utilizes year in which movie was released. This is further divided into into periods; 1940’s, 1950’s etc.
- **Movie Lens dataset**

The first dataset is provided by the MovieLens platform, a sub company of the groupLens project. It has provides data on 100,000 ratings and 1,300 tags applied to 9,000 movies by 700 users. The data is structured into files which are written as comma-separated values files with a single header row. The files are encoded as UTF-8 and the system has to be aware of the fact, so that there is no issues when reading the files.

The file, which is of utmost importance for this project is the Ratings Data file. Re:Voire will make suggestions to users of what they could watch next, using collaborative filtering. For this a data where users and their ratings are defined is necessary for a recommender system. All ratings are contained in the ratings.csv. Each line of the file after the header row represents one rating of one movie by one user. The lines within the file are ordered first by userId, then within user by movieID. The ratings in the data are consistent with the ones from the website, where the rating system ranges from 0.5 stars (worst) to 5 stars (best) in the 10 point scale. The last column in the file refers to the Timestamp, which represents seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970 (Harper, & Konstan, 2016).

Out[43]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

Figure 12: Ratings data

MovieLens users where selected at random for inclusion. Their ids have been anonymized and are consistent between the other files in the dataset e.g. the same id refers to the same user in the other files. The movie items included in the dataset have at least one rating from the users. These movie ids are presented with those used on the MovieLens website and are comprised between the other files All tags included in the dataset are user-generated metadata about each movie. Each tag is typically a single word or short phrase. The meaning value and purpose of a particular tag is determined by each user. The tags are contained in the tags.csv, where each line of this file after the header row represents one tag applied to one movie by one user (Harper, & Konstan, 2016).

The information for each movie is contained in the movies.csv. Each line of this file after the header row represents one movie, the id of the movie, title and genres. Movie titles are entered manually or imported from themoviedb.org (tMDB) and include the year of release in parentheses. Errors and inconsistencies may exist in the titles because of that. The genres are a pipe-separated list, and are selected from the most used ones in tMDB:

Action / Adventure / Animation / Children's / Comedy / Crime / Documentary / Drama / Fantasy / Film-Noir / Horror / Musical / Mystery / Romance / Sci-Fi / Thriller / War / Western / (no genres listed)

	movieId	title	genres	year
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II (1995)	Comedy	1995

Figure 13:: MovieLens Movie Data

The dataset also includes a file called links that links the movie IDs from the MovieLens website with the IDs of the most popular databases themoviedb.org and imdb.com in a format where each row represents one movie ID with its corresponding ID in the other databases.

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

Figure 14: Movie Links Data

- **TMDB 5000 Movie Dataset**

provided by Kaggle. The dataset contains a set of 5000 films from The Movie Database (tMDB) in accordance with their terms of use. The dataset is divided into two subsets Movies (containing the movies and their additional information such as overview, tagline and so on. The second subset is composed by the movie credits, like actors and directors). The dataset contains a separate file that contains full credits for both the cast and the crew. All fields are filled out by users so it is not to be expected to agree on keywords, genres, ratings, or the like. Several of the columns in the dataset contain information that is in the json format. Kaggle provides a kernel with a tutorial for reading the json format and implementing it into a live python notebook format. The features of each movie is represented by the following information:

- homepage
- id
- original_title
- overview

- popularity
- production_companies
- production_countries
- release_date
- spoken_languages
- status
- tagline
- vote_average

The dataset has been generated from tMDB API, but is not endorsed or certified.The API provides access to data on many additional movies, actors and actresses, crew members and TV shows.

6.3 Bridge

For the Re:voir recommendation system to work it is important to understand what metadata is missing in the current dataset, what data was used in our prototype. By looking at the table below we can see that; Firstly, current Re:voir film metadata doesn't have a section for genres, in fact, they don't have a unified structure to categorize films into genres, so those genres which appear in their dataset is treated as keywords. They would need to categorize each of the films in their catalogue in genres provided in the Experimental Film section. Secondly, create a database for users, so that information about their film ratings and viewing behaviour could be stored. Furthermore, giving each film an ID would bypass issues with titles being in different languages for different regions. Lastly, there is no need for the summary section as new user interface design would support only one entry due to the user experience.

Re:voir(current)	**MLens**	**TMDB**	**Re:voir(new)**
Title	Title	Title	Title
Author		Director	Author
Summary		Tagline	
Description		Overview	Description
Keywords	Tags	Keywords	Keywords
Year	Year	Year	Year
	Genres	Genres	Genres
	User rating		User rating
	MovieID	MovieID	MovieID

Table 2: Connection between the different data

The table below shows relation between metadata, search engine row categorization and algorithms that would drive the recommendation system.

Revoir	**Search engine**	**Categorization**	**Collaborative Filtering**	**TF/ID**	**Count Vector**
Title					
Author					
Summary					
Year					
Keywords					
Genres					
User rating					
MovieID					

Table 3: Metadata And Algorithm Relation

Thumbnail images of film posters should be linked to each of the MovieID's. Lastly, images of the author and short biography should be linked to each of the authors so this information could automatically appear in the user interface.

6.4 Data Analysis and Development

- **Tools used for developing**

Python

The system is built using the Python 3.# programming language. Python is a general-purpose language, which means it can be used to build a program using its powerful and easy to use tools and libraries. Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code reliability. The syntax in python helps the development process to do coding in fewer steps as compared to other languages. (Mediafire, 2017) The language is widely used in bigger organisations, because of its multiple programming paradigms. The language is compatible with a comprehensive and large standard library that has automatic memory management and dynamic features, necessary for the development of this project. On a professional level, the programming language has proven, over the years, to be a great choice for data analysis, machine learning development and scientific computing. It is

easy to understand and very flexible. Another key aspect for choosing Python as our development language is its easy web integration. This will ensure the easy port of the system to the Re:Voir web application in the future.

IBM Data Science Experience

The system is built using IBM Data Science Experience (DSX) platform and cognitive class labs (CC Labs), which provides everything necessary for developing a project such as this one. The platform saves lot of time and effort for data analysis to relocate time toward more valuable tasks such as the actual analysis of data instead of environment readiness. The most valuable feature of the DSX is that it is a platform that unifies numerous open source components. Some of the greatest challenges in the data analysing field includes deployment, maintenance etc. but DSX includes everything necessary for such endeavours, saving valuable time and resources. The Data science experience platform operates on the cloud, with a client that can be accessed with any Internet web browser.

DSX includes the Anaconda Distribution as an architecture, ("the easiest way to do Python data science and machine learning") with one key element. The distribution acts as an architecture which serves as a base for building any type of python projects from the ground up. It also comes with almost all the necessary libraries to develop projects such as this one. Other platforms that may have been used, include the likes of Kaggle which also operates on the could and provides a more stable Kernel and a peer sharing setup which provides the ability to easily share documents between peers. But despite it providing slightly more stable architecture it is missing crucial elements and libraries that are needed for this project.

Project Jupyter

The Jupyter notebook is an open-source web application that allows the development and sharing of documents that contain live code, equations, visualizations and narrative text. It is included within the Anaconda distribution with a lot of different tools which are helpful for any type of development. The uses of the notebook include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning etc. The Notebook supports over 40 programming languages, including the most popular data science languages, like Python, R, Julia. This project is built with the JupyterLab Integrated development environment (IDE) which creates a better experience for managing information. The live code provided in JupyterLab is easy to test out and pinpoints the mistakes on the fly, giving a great code management environment.

6.5 System Development

To start things off, the project needs to initialise the libraries necessary to handle the data and code for the program. All the libraries and tools used are open-sourced and provide high performance. They are developed to be used when programming under the python architecture and can be imported using the Anaconda Distribution. The most important libraries are:

Pandas is a library is developed to easily use data structures and data analysis tools for the Python programming language.

NumPy is another important library that contains the fundamental package for scientific computing with Python. It contains a powerful N-dimentional array objects and useful linear algebra tools among other.

Scikit-learn is another popular data science library which has simple and efficient tools for data mining and data analysis. These tools are built on the above mentioned NumPy SciPy and matplotlib (a library which utilises the representation of data visually)

It has been estimated that the Re:Voir database will not be sufficient in the development and testing of the algorithms for the system. That is why, the system will be using two open source datasets instead of the Re:Voir database. They contain movies with additional information that mimics the data for experimental movies and User ratings information that will mimic users for Re:Voir.

MovieLens Latest Dataset 100k contains 100,000 ratings applied to 9,000 movies by 700 users. It is the most current movie dataset available. The recommender system will use the ratings of the users as a base for predicting weather or not a given movie will be rated (Liked/disliked).

TMDB 5000 dataset data will be imported and into a data-frame using the Pandas library. The system needs to make personalised predictions, therefore the data that the system will use has to be linked with the MovieLens dataset. MovieLens have provided a table linking all the necessary movieIDs from different databases, which include IMDB indexes and TMDB indexes. Since the system will work with the TMDB dataset, it will include the indexes from MovieLens and TMDB.

Out[25]:

	0
budget	2.37e+08
genres	[{"id": 28, "name": "Action"}, {"id": 12, "nam...
homepage	http://www.avatarmovie.com/
id	19995
keywords	[{"id": 1463, "name": "culture clash"}, {"id":...
original_language	en
original_title	Avatar
overview	In the 22nd century, a paraplegic Marine is di...
popularity	150.438
production_companies	[{"name": "Ingenious Film Partners", "id": 289...
production_countries	[{"iso_3166_1": "US", "name": "United States o...
release_date	2009-12-10
revenue	2.78797e+09
runtime	162
spoken_languages	[{"iso_639_1": "en", "name": "English"}, {"iso...
status	Released
tagline	Enter the World of Pandora.
title	Avatar
vote_average	7.2
vote_count	11800
cast	[{"cast_id": 242, "character": "Jake Sully", "...
crew	[{"credit_id": "52fe48009251416c750aca23", "de...

Figure 15: The dataset used for developing the recommender system.

There are 20 categories inside the movie data, which are going to be considered as features for our Recommendation system. To avoid confusion further on, some of the unnecessary features will be dropped, and both the movie data and the credits data will be merged together to form one data-frame. The most important features for this system which will be kept are Title, Overview and Taglines, Cast, Keywords and Genre, as these features are the closest resemblance for the Experimental Movie database.

Overview refers to the description of a certain movie, and Tagline is the equivalent to the Summary in the Re:Voire database. Cast contains the director, editor and other members of each movie, but only the Director will be used further on, since he is the equivalent to the to the Author. The other features are considered to be the same. The new data-frame contains 4803 movies with now 8 columns of information, which will be used as feature vectors for our Recommender system.

6.5.1 Content-based recommender system

The first step for building the recommender system is to build an engine that computes the similarity between movies based on certain criteria. It should then suggest movies that are similar to the active movie which the user is currently watching. The first instance of the the Recommender system uses the movie Overview and Taglines (from the content wrangled above) to produce the prediction. This will be judged qualitatively, with the use of the TF-IDF technique, since there is no quantitative metric in the dataset. SciKit-learn contains a tool for this technique which eases the whole process.

```
In [27]: md['tagline'] = md['tagline'].fillna('')
         md['description'] = md['overview'] + md['tagline']
         md['description'] = md['description'].fillna('')

In [28]: tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
         tfidf_matrix = tf.fit_transform(md['description'])
         tfidf_matrix.shape

Out[28]: (4803, 110167)
```

Figure 16: TF-IDF Initialisation

After the TF-IDF technique has produced the term weight vectors, the system needs a similarity measure for describing the proximity of two weighted vectors. The Cosine similarity measure will produce a numeric quantity that denotes the similarity between two movies. Also, after calculating the dot product from the TF-IDF Vectorizer function we will have a cosine similarity score. The Linear_kernel function from Sklearn will produce a much faster result.

```
In [29]: cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
         cosine_sim[0]

Out[29]: array([1.        , 0.00528077, 0.        , ..., 0.        , 0.        ,
                0.        ])
```

Figure 17: Cosine Similarity

The code above has produced a cosine similarity matrix for all movies. Next, the system needs a function that extracts the current information from the similarity measure and produces a list with the most similar movies based on the score. This function can be used to generate the recommendations. As stated the system will work based on the current movie being watched. For simplicity the system uses the title, rather than the id of movies, as this is easier to test and understand. The calling of the function below with the name of the movie "The Dark Knight" will give the 10 most relevant movies according to the Overview and Tagline of the movies.

```
In [31]: get_recommendations('The Dark Knight').head(10)

Out[31]: 3                          The Dark Knight Rises
         290                               Batman Forever
         412                               Batman Returns
         2990    Batman: The Dark Knight Returns, Part 2
         1251                                      Batman
         1102                                         JFK
         117                               Batman Begins
         827                         Law Abiding Citizen
         198           Sherlock Holmes: A Game of Shadows
         9             Batman v Superman: Dawn of Justice
         Name: title, dtype: object
```

Figure 18: Recommendation based on Description

So far the system generates good predictions based on the movie description that has been used. This prediction engine is equivalent to the one the Re:Voire system will use to generate its movie recommendations with its experimental movies. But it is still incomplete, since it doesn't use the other features of the data-set. The system needs to strips spaces and convert the characters into lower cases for the director's name, otherwise the engine will consider Steven Spielberg and Steven Soderbergh as the same person, confusing the algorithm. The keywords feature, also needs to be preprocessed before it can be used.

```
Out[88]: duringcreditsstinger    257
         woman director          198
         independent film        193
         based on novel          161
         aftercreditsstinger     147
         Name: keyword, dtype: int64
```

Figure 19: Keywords count

The frequency counts of the keywords that are included in the dataset, with the least instances occurring once and the most frequent 324 times. In a recommender system, words that occur only once are not useful for recommendations. Also, some words are such as Bird and Birds are considered with the same meaning, the only difference is that one is plural. The system cannot distinguish this so it needs to consider them the same.

The next step is to generate the recommendations. This will be done slightly different than the previous approach. The engine will put all of the features for every movie in one metadump. The Count Vectoriser tool from Scikit-learn will then create a count matrix for this metadump, like the one used before with the TF-IDF technique. After that the Cosine Similarity measure will help generate the most similar movies recommendation of a current movie. Now the engine is ready to generate recommendations. The function

initiated earlier can be reused here. The following output will generate predictions using the Keywords, Author and Genres.

```
In [42]: get_recommendations('The Dark Knight').head(10)

Out[42]: 3                    The Dark Knight Rises
         117                          Batman Begins
         1117                          The Prestige
         95                               Inception
         963                               Insomnia
         2838                               Memento
         94                            Interstellar
         412                         Batman Returns
         203                         Batman & Robin
         9       Batman v Superman: Dawn of Justice
         Name: title, dtype: object
```

Figure 20: Recommendation on Keywords, Author and Genres

The recommendations here are different than the previous ones, and possibly a little more accurate, because of the different data used. The Author has made a great impact on the recommendations as Christopher Nolan's movies are more dominant in this situation. In the field of Experimental cinema, the Author and his touch is one of the most important aspect people expect to see when viewing experimental cinema, so this is taken into consideration in the example.

This recommendation system only considers movies as a means to suggest items. In short, the system doesn't capture personal opinions and biases, so anyone who watches movies will have the same suggestions. This is a major flaw for the ultimate goal. Most systems utilise the Users and their ratings, hence the user input needs to be included. By using this information, the system will be capable of capturing the tastes of the users and generate personal recommendations for each individual user.

6.5.2 Collaborative Filtering

As mentioned earlier in the project, the Collaborative filtering approach is based on the idea that similar users can be used to predict if a movie is going to be liked by another similar user who has never seen the item. Hence the recommendation will predict ratings based on how other users have rated the movie. The data required for this is taken from the MovieLens dataset of users that contains 100,000 ratings from 700 users on 9000 movies. Assuming that Re:Voire will use a similar database for users who rate movies the same way the MovieLens users have rated popular feature movies.

This approach will be implemented first as a stand alone feature and after that, it will be combined with the previous engine to create a hybrid recommendation system. The Collaborative Filtering will be handled with the powerful Singular Value Decomposition from the Surprise library, in order to minimise the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE).

Out[43]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

Figure 21: User Data

The image shows the users and their ratings from the MovieLens subset. It is consisted of the IDs of each user, the movieID for each movie rated by the user and the ratings given from the scale 10 point scale of 0.5 to 5 with 2.5 being the middle. First the data will be divided using the 5 fold cross validation model. This will ensure that the SVD model will generate a more precise prediction further on.

```
Evaluating RMSE, MAE of algorithm SVD.

------------
Fold 1
RMSE: 0.8915
MAE:  0.6873
------------
Fold 2
RMSE: 0.8940
MAE:  0.6904
------------
Fold 3
RMSE: 0.9043
MAE:  0.6947
------------
Fold 4
RMSE: 0.9103
MAE:  0.7018
------------
Fold 5
RMSE: 0.8815
MAE:  0.6778
------------
------------
Mean RMSE: 0.8963
Mean MAE : 0.6904
------------
------------
```

Figure 22: RMSE & MAE Scores

the MAE is 0.69, which in theory has to be lower than the RMSE. The RMSE is 0.89 and which seems to be enough for the needs of the system. These values ensure that the the recommender system works as expected. After we have optimised the SVD algorithm, the recommender system needs to make a training

set in order for the predictions to be generated. The training set contains the decomposed values. Now the Surprise toolset can generate predictions based on the training data that was injected. The prediction model works by picking a user who has already rated some movies, and picking a random movie. The system takes this as input and will generate a prediction on the score based on the estimates in the training set. The score is based on the range between 1 to 5 as in the movie ratings dataset In simple terms, the score determines whether the user will like or dislike the movie. The function which will be used will generate the predictions, where 1 is associated with the user ID and 55 corresponds to the movie ID. The result is given from the SVD predictor and can be interpreted as, user 1 has rated movie 55 with an estimated score of 2.46. In other words, the system predicts that the user will rate movie 55 with this score.

```
In [47]: svd.predict(1, 55, 3)

Out[47]: Prediction(uid=1, iid=55, r_ui=3, est=2.461484480114258, details={'was_impos
         sible': False})
```

Figure 23: Collaborative Filtering Rating

6.5.3 Hybrid System

As mentioned before this Collaborative filtering model as it stands, works only to estimate scores. It doesn't take into consideration what the movie is, or what it contains. This is definitely not enough for a proper movie recommendation system to work. Combining both methods together to create a hybrid system is essential. This hybrid system will use the aspects that what we have developed so far to generate personalised recommendations. If we consider this to be the Re:Voire system, it will be able to recommend experimental films which it is estimated that individual users will like, based on the information given in the context of the movie and the other users ratings. The system will take UserID and Movie Title as inputs and will output Similar movies sorted on the basis of expected ratings.

After setting it up, the system can now generate predictions based on User input (ratings) and the context of movies. The recommendation for each user consists of 10 movies compiled from the content and grouped by their estimated ratings in a descending order. The following example shows the predictions for two people who are watching the same movie, but have slightly different recommended movies based on the predicted score they are assumed to give. After this the system will be tested on real users and their results will be considered as a validation to the system's ability to generate predictions.

```
In [53]: hybrid(1, 'Avatar')
```

Out[53]:

	title	id	est
270	Terminator 2: Judgment Day	280	3.254327
2747	The Terminator	218	3.112695
2063	Aliens	679	3.094833
46	X-Men: Days of Future Past	127585	3.094260
47	Star Trek Into Darkness	54138	2.879391
2011	Predator	106	2.825475
819	Superman II	8536	2.733227
1904	The Covenant	9954	2.725979
4	John Carter	49529	2.679809
72	Suicide Squad	297761	2.629352

```
In [74]: hybrid (7, 'Avatar')
```

Out[74]:

	title	id	est
2747	The Terminator	218	3.984383
2063	Aliens	679	3.737076
47	Star Trek Into Darkness	54138	3.552689
270	Terminator 2: Judgment Day	280	3.543133
2011	Predator	106	3.500529
556	The Abyss	2756	3.498759
1904	The Covenant	9954	3.357531
46	X-Men: Days of Future Past	127585	3.356254
510	Titan A.E.	7450	3.275952
251	Ender's Game	80274	3.251230

Figure 24: Hybrid System Recommendations

The top row of the tables consist of the index numbers, Title of the movie recommended, the ID number of the movie, and the estimated score the system has predicted. The score is in the range of 0.5 to 5. When implementing the system, these scores will resemble likes or dislikes, anything above a threshold of 2.5 will be considered a “like” and anything below that will be considered a “dislike”.

6.6 Evaluation

- **Methodology**

After the system has been developed, it is important to evaluate its credibility. When designing the evaluation method, three different elements had to be taken into consideration. First, how the system generates personalised predictions of items and how they are presented. Second, how this information is stored in the dataset and how it is retrieved. Lastly, since the system revolves around experimental films, it needs to be credible in its field. These three factors were the base for our evaluation test. In order to be more credible, the system was tested on film major students in Dongguk University Seoul (South Korea). The students that the test was conducted on are considered new users for the system, which mimics the process when new users start using the system. This provided vital information for testing the system’s ability to predict movies.

The test was conducted using a two stage questionnaire (with the use of Google forms) where the first stage was created as a tool to bypass the Cold-Start problem. In this stage the new users have to rate with “stars” a number of different movies, included in the movie dataset used to train the system. The rating margin was set up with values scale of 1 to 10, where 1 means the user does not like the movie and 10 means the user loves the movie. This corresponded with the MovieLens rating of (0.5 to 5). For example, when a user rates a movie from the questionnaire with a value of 5, it corresponds to 2.5 in the dataset used for developing this project. It is important to note that the rating process is determined on weather people have watched a particular movie. In other words, if users have not seen a particular movie they would not give it a rating. The movies were hand picked at random from the tMDB top movies chart,

which ensures that more users will rate more movies, further eliminating the cold start problem. The data was stored anonymously, but their contacts were kept for the second stage.

At the end of the questionnaire each user was asked to give their favorite movie. This was necessary to our system, which was considered to resemble the movie that the user is currently watching. This favorite movie will be implemented as the input in the recommendation algorithm. After the users have completed the first questionnaire, and rated enough movies, the results were implemented in the Ratings dataset, with the correct values. The system was retrained with the current values and was re enabled to generate personalised predictions for those users. The predictions were the same format as the results in the development phase, where the system generated a prediction of 10 movies with their estimated ratings, indicating that the users were going to like or dislike the movies. With this the first stage was concluded and the results were documented.

The second stage was designed to compare the personalised predictions with the user`s expected ratings. Users were given a personalised questionnaire (with the use of Google Forms) that was tailored according to their personalised movies generated by the system. The users were asked to rate those movies on the scale of 1 to 10, where 1 means the user does not like the movie and 10 means the user loves the movie. After the results were gathered the comparison between the ratings estimated by the system and the ratings given to the movies was made possible. If the system and the users have approximately the same ratings, then the prototype can continue to its next stage. If the approximation was off by a large margin, the system will be considered flawed and needs to be reworked, using different methods, and further reevaluated.

- **Results.**

The test was handed out to a total of 45 students, but 18 participants who all gave a rating score to at least one movie were documented. The movie catalog from which the participants were able to give their ratings consisted of 15 popular feature movies like “The Dark Knight (2008)” or “Oldboy (2003)” to name a few. The movie with least ratings was “Mission Impossible (1996)” and the movie with the highest number of ratings were “Toy Story (1995)” and “Spirited Away (2001)”.

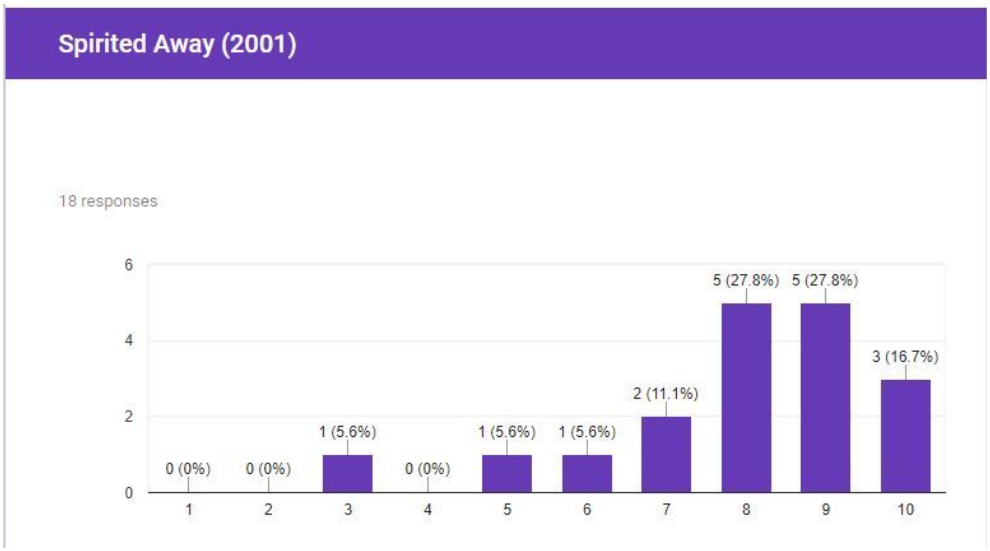

Figure 25: "Spirited Away"

The ratings of each user were converted to resemble the score scheme in the dataset and were implemented as instances inside it. The system was retrained with the new instances and predictions for each user were made, using the user index (number) and the name of their favorite movie as input. The following example indicates a comparison between the prediction results for two test user after the system has generated its prediction in this stage.

```
In [29]: hybrid (672, 'Forrest Gump')
```

Out[29]:

	title	id	est
1981	Back to the Future	105	4.561516
1331	The Walk	285783	4.280796
2647	Woman on Top	14629	4.168299
2234	You Will Meet a Tall Dark Stranger	38031	4.084098
2042	Being Julia	18701	4.037255
369	Cast Away	8358	4.025547
60	A Christmas Carol	17979	4.016743
1394	Flight	87502	4.009525
2734	The Four Seasons	25113	3.959741
310	Sex and the City 2	37786	3.935723

```
In [30]: hybrid (680, 'Cloud Atlas')
```

Out[30]:

	title	id	est
3164	Run Lola Run	104	4.005998
2875	Moon	17431	3.900190
2397	About Time	122906	3.851203
2240	Metropolis	19	3.721862
2575	Melancholia	62215	3.566089
2402	Fortress	12088	3.517653
483	Children of Men	9693	3.509579
2235	Never Let Me Go	42188	3.503697
3371	Another Earth	55420	3.466671
3095	Sleep Dealer	20764	3.416534

Figure 26: Comparison between two users

The questionnaire for the second stage was developed with these results. Each user had personalised questionnaires, and was then asked to rate the predicted movies using the same process and scale as the previous. The results from the second stage were gathered and a comparison between the system`s

prediction and the actual rating was conducted. The most notable comparison that could be addressed, is the downward trend of ratings given to the predicted movies. The following graph shows the results of 5 users after the second stage, indicating the downward trend.

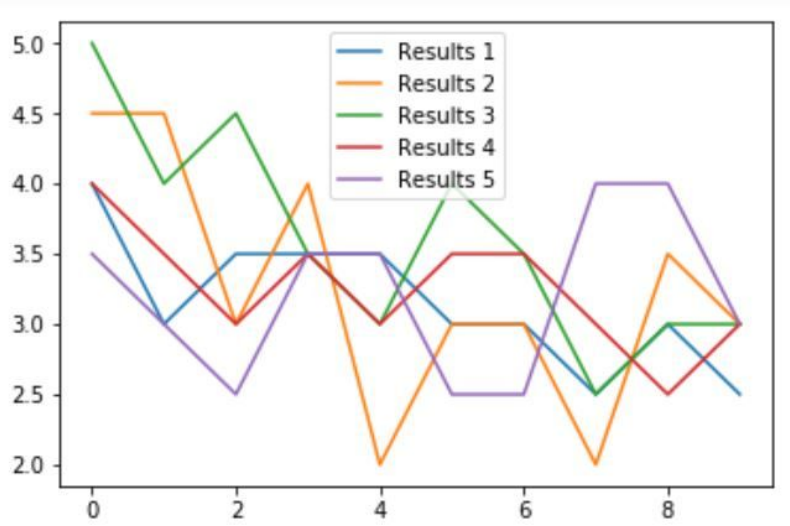

Figure 27: Result Trends

The following figure shows the results of one user`s ratings compared with the rating score generated by the system:

```
In [34]: hybrid (675, 'V for Vendetta')
```

Out[34]:

	title	id	est
1964	Equilibrium	7299	3.991769
268	Casino Royale	36557	3.928893
432	The Book of Eli	20504	3.829294
331	Unstoppable	44048	3.737812
1194	The Crow	9495	3.713286
1178	Cirque du Freak: The Vampire's Assistant	24418	3.694580
1900	Daybreakers	19901	3.663902
3139	Dinner Rush	22617	3.597281
1970	Survivor	334074	3.588031
1957	The Cold Light of Day	77948	3.585859

	Given result	Title
0	4.0	Equilibrium
1	3.0	Casino Royale
2	3.5	The Book Of Eli
3	3.5	Unstopapble
4	3.5	The Crow
5	3.0	Cirque du Freak: The Vampires Assistant
6	3.0	Daybreakers
7	2.5	Dinner Rush
8	3.0	Survivor
9	2.5	The Cold Light of Day

Figure 28: Comparing both predicted and given results

7 Discussion

- **Trends**

The biggest challenge for the company is to generate a narrowed down approach for figuring out their needs and generating a plan for development. As mentioned, the objective is to expand their business towards the current technological trends and maintain a new vision to distinguish themselves as the current go to service provider for experimental film streaming. In order to achieve this the company needs to have a good understanding of the current technological trends set by the big companies who are a leader in their field (Netflix for feature film streaming on demand, Spotify for music streaming, Youtube as Video recommendation service to name a few). This will make the company more cautious on how to tackle their problems and how to expand even further in the future.

- **Technology**

Generating a service that provides video on demand to mobile or other platforms is not an easy task. The main focus of this project was to update the database with a way to properly classify experimental films. This classification will be used as a base for developing a system that generates predictions. As the company's current database needs to be reworked, the system had to be developed using a dataset provided by other internet movie databases, like MovieLens and The movie database. This database was wrangled to resemble the vision of the new classification and the current database of Re:Voir, so that it can produce the same results.

The new system was developed using the current technology, through the python development language for easy integration in the future. This enabled the idea of the concept to be developed without any issues outside of the normal. The prediction system worked as intended, suggesting movies, with a predicted rating, but the results were not evaluated with the right method. This created an issue for the system, because its usefulness was not evaluated properly. This drawback will have to be addressed in the future implementation of the recommender system.

- **Issues with the test**

The results from the test were positive, but it goes without saying that the method used to evaluate the system was not efficient. The most notable issue with this evaluation test was the fact that not everyone participated, which gave a non-reliable result. The second major issue has to do with the fact that the users did not rate all the movies to eliminate the cold-start problem. The prediction with some users seemed to be non consistent, which lead to a mismatch from the general downward trend of ratings (see Figure #, Result 5). If the users were able to rate more movies, the system would have been refined with a more consistent and probably accurate result. Another problem with this test was the way it was designed. It was difficult to keep track of the results in the second phase, which was time consuming to generate each individual questionnaire.

The paradox that may be taken as a lesson from here comes from the problem that most users did not watch the movies that were suggested to them. This proved that the system did what it was supposed to do, predict items to the users that they may have never seen before. But this was an issue when interpreting the results, the fact that most users did not rate some movies gave a sparse data that was not used in the analysis. This is due to the poor design of the evaluation method that needs to be addressed when considering to reevaluate the system.

- **Future development**

As mentioned in the future the evaluation method needs to be reconsidered and redone, so that the system is justified as performing as it should. But the development of the system with the cutting edge technology would be of essence. If the company needs to stand out in the market, providing its platform on par with Netflix or Youtube, it needs to start viewing the possibilities for further developing their product.

A possible way to go forward is to use Semantic Analysis as a base for analysing the abstractness of the movies` descriptions, so that the system can understand the aspects of the text more in detail. This will enable more accurate and objective suggestions for users, using explicit data. Another aspect is to start implementing ways to gather implicit data from users. This implicit data could help create more personalised accounts that capture the user's interests. This data could then be analysed through neural networks as a method to predict and suggest items with more details. This will make customers use the application more, which in turn will generate more data, entering the holy grail stage of data analysis.

8 Conclusion

As a company, Re:Voir is an experienced entity, which knows how to manage their problems. With their more than 20 years of experience in the field, it is clear to them what they need and what steps they should attend in order for them to move forward. As the current trends in the industry are all converging towards managing big data, it is without a doubt that this is the direction they should go forward with. The updating of their database system should be made easy with the current research and it should be clear of the procedure. The main purpose of this project was to ease the process behind this update by classifying experimental films into new categories, by which the company would expand their line of experimental films and would continue to develop further. With this classification it would be easier to import a new data analysing system. With a newly developed application for mobile, Re:Voir would have the possibility to enlarge its pool of customers and provide a new service with an on demand video streaming platform. This platform has to be developed with the current technology so that the whole experience meets the current trends. The system will include a recommender engine, using the current technology that suggests new movies to its audience. This will in turn make customers stay on the platform more, generating implicit data that would then be used to further improve the experience. In the end this will create a loop where customers generate data and the company will improve its system to make their users generate more data. In an ideal world this is the goal for every company to become an unstoppable force in the field in which it operates.

9 References

1. 4.0.0.1 About Re:Voir - REVOIR. (2018). Re-voir.com. Retrieved from https://re-voir.com/shop/en/content/7-about-revoir

2. Peterson, J. (1994). *Dreams of Chaos, Visions of Order: Understanding the American Avante-garde Cinema*. Wayne State University Press.

3. Sitney, P. A. (2002). *Visionary film: the American avant-garde, 1943-2000*. Oxford University Press, USA.

4. Rollins, J. (2015). *Foundational Methodology for Data Science*. Slideshare.net Available at: https://www.slideshare.net/JohnBRollinsPhD/foundational-methodology-for-data-science.

5. Bruni, L. E., & Baceviciute, S. (2013, November). Narrative intelligibility and closure in interactive systems. In *International Conference on Interactive Digital Storytelling* (pp. 13-24). Springer, Cham.

6. Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *5*(4), 19.

7. Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 191-198). ACM.

8. Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008, June). Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Applications in Management* (pp. 337-348). Springer, Berlin, Heidelberg.

9. Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., ... & Sampath, D. (2010, September). The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 293-296). ACM.

10. Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008, October). Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on Recommender systems* (pp. 267-274). ACM.

11. Gomez-Uribe, C. A., & Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, *6*(4), 13.

12. Bawden, D., & Robinson, L. (2009). The dark side of information: overload, anxiety and other paradoxes and pathologies. *Journal of information science*, *35*(2), 180-191.

13. Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-based systems*, *46*, 109-132.

14. Adomavicius, G., & Tuzhilin, A. (2015). Context-aware recommender systems. In *Recommender systems handbook*(pp. 191-226). Springer, Boston, MA.

15. Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1-35). springer US.

16. Amatriain, X., & Pujol, J. M. (2015). Data mining methods for recommender systems. In *Recommender systems handbook*(pp. 227-262). Springer, Boston, MA.

17. Koren, Y., & Bell, R. (2015). Advances in collaborative filtering. In *Recommender systems handbook* (pp. 77-118). Springer, Boston, MA.

18. Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73-105). Springer, Boston, MA.

19. Choi, S. M., Ko, S. K., & Han, Y. S. (2012). A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, *39*(9), 8079-8085.

20. Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook* (pp. 107-144). Springer, Boston, MA.

21. Barza, S., & Memari, M. (2014). Movie genre preference and culture. *Procedia-Social and Behavioral Sciences*, *98*, 363-368.

22. (Developing the movie genres based on culture)

23. Nath, K., Dhar, S., & Basishtha, S. (2014, February). Web 1.0 to Web 3.0-Evolution of the Web and its various challenges. In *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on* (pp. 86-89). IEEE.

24. Billsus, D., & Pazzani, M. J. (1998, July). Learning Collaborative Information Filters. In *Icml* (Vol. 98, pp. 46-54).

25. Koohi, H., & Kiani, K. (2017). A new method to find neighbor users that improves the performance of Collaborative Filtering. *Expert Systems with Applications*, *83*, 30-39.

26. Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, *7*(1), 76-80.

27. 4.0.0.2 Introduction to Recommender Systems: A 4-hour lecture. (2014). Technocalifornia.blogspot.bg. Retrieved 27 April 2018, from www.technocalifornia.blogspot.bg http://technocalifornia.blogspot.bg/2014/08/introduction-to-recommender-systems-4.html?m=1&from=singlemessage&isappinstalled=0

28. Schach, S. R. (2002). Object-oriented and classical software engineering (Vol. 6). New York: McGrawHill.

29. Cockburn, A. (2006). Agile software development: the cooperative game. Pearson Education

30. Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *5*(4), 19.

31. 4.0.0.3 Mediafire (2017) Advantages and Disadvantages of Python Programming Language. Medium. Retrieved from https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

The System

Useful platforms

The system is built using the Python 3.# programming language. Python is a general-purpose language, which means it can be used to build a program using its powerful and easy to use tools and libraries. On a professional level, the programming language has proven, over the years, to be a great choice for data analysis, machine learning development and scientific computing. It is easy to understand and very flexible. Another key aspect for choosing Python as our development language is its easy web integration. This will ensure the easy port of the system to the Re:Voire web application in the future.

The system is built using IBM Data Science Experience platform (CC Labs), which provides everything necessary for developing a project such as this one. It includes the Anaconda Distribution as an architecture, ("the easiest way to do Python data science and machine learning") with one key element. It includes everything necessary to develop projects such as this one. The Data science experience platform operates on the cloud, with a client that can be accessed with any Internet web browser. The applications Other platforms that may have been used, include the likes of Kaggle which also operates on the could and provides a more stable Kernel and a peer sharing setup which provides the ability to easily share documents between peers. But despite it providing slightly more stable architecture it is missing crucial elements and libraries that are needed for this project.

The Jupyter notebook is an open-source web application that allows the development and sharing of documents that contain live code, equations, visualizations and narrative text. The uses of the notebook include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning etc. The Notebook supports over 40 programming languages, including the most popular data science languages, like Python, R, Julia. This project is built with the JupyterLab Integrated development environment (IDE) which creates a better experience for managing information. The live code

Building the system

To start things off, the project needs to initialise the libraries necessary to handle the data and code for the program. All the libraries and tools used are open-sourced and provide high performance. The most important libraries are:

Pandas is a library is developed to easily use data structures and data analysis tools for the Python programming language.

NumPy is another important library that contains the fundamental package for scientific computing with Python. It contains a powerful N-dimentional array objects and useful linear algebra tools among other.

Scikit-learn is another popular data science library which has simple and efficient tools for data mining and data analysis. These tools are built on the above mentioned NumPy SciPy and matplotlib (a library which utilises the representation of data visually)

In [24]:

```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD, evaluate

import warnings; warnings.simplefilter('ignore')
```

As stated, the system will be using two open source datasets instead of the Re:Voire database. They contain movies with additional information that mimics the data for experimental movies and User ratings information that will mimic users for Re:Voire.

MovieLens Latest Dataset 100k contains 100,000 ratings applied to 9,000 movies by 700 users. It is the most current movie dataset available. The reccomender system will use the ratings of the users as a base for predicting weather or not a given movie will be rated (Liked/disliked).

TMDB 5000 dataset provided by Kaggle. The dataset contains a set of 5000 films from The Movie Database (tMDB) in accordance with their terms of use. The dataset is divided into two subsets Movies (containing the movies and their additional information such as overview, tagline and so on. The second subset is composed by the movie credits, like actors and directors). It has been generated from tMDB API, but is not endorsed or certified. The API provides access to data on many additional movies, actors and actresses, crew members and TV shows. The information provided includes everything needed for the goals of this project.

First the tMDB movie data will be imported and into a data-frame using the Pandas library. The system needs to make personalised predictions, therefore the data that the system will use has to be linked with the MovieLens dataset. MovieLens have provided a table linking all the necessary movieIDs from different databases, which include IMDB indexes and TMDB indexes. Since the system will work with the TMDB dataset, it will include the indexes from MovieLens and TMDB.

In [25]:

```
md = pd.read_csv('tmdb_5000_movies.csv')

links_small = pd.read_csv('../links.csv')
links_small = links_small[links_small['tmdbId'].notnull()]['tmdbId'].astype('int')

md['id'] = md['id'].astype('int')
md = md[md['id'].isin(links_small)]
md.shape

cd = pd.read_csv('tmdb_5000_credits.csv')
cd = cd.drop('title',axis=1)
cd = cd.rename(columns = {'movie_id':'id'})

md['id'] = md['id'].astype('int')
cd['id'] = cd['id'].astype('int')

md=pd.merge(md,cd,on='id',how='outer')
md.head(1).transpose()
```

Out[25]:

	0
budget	2.37e+08
genres	[{"id": 28, "name": "Action"}, {"id": 12, "nam...
homepage	http://www.avatarmovie.com/
id	19995
keywords	[{"id": 1463, "name": "culture clash"}, {"id":...
original_language	en
original_title	Avatar
overview	In the 22nd century, a paraplegic Marine is di...
popularity	150.438
production_companies	[{"name": "Ingenious Film Partners", "id": 289...
production_countries	[{"iso_3166_1": "US", "name": "United States o...
release_date	2009-12-10
revenue	2.78797e+09
runtime	162
spoken_languages	[{"iso_639_1": "en", "name": "English"}, {"iso...
status	Released
tagline	Enter the World of Pandora.
title	Avatar
vote_average	7.2
vote_count	11800
cast	[{"cast_id": 242, "character": "Jake Sully", "...
crew	[{"credit_id": "52fe48009251416c750aca23", "de...

There are 20 categories inside the movie data, which are going to be considered as features for our Recommendation system. To avoid confusion further on, some of the unnecessary features will be dropped, and both the movie data and the credits data will be merged together to form one data-frame. The most important features for this system which will be kept are Title, Overview and Taglines, Cast, Keywords and Genre, as these features are the closest resemblance for the Experimental Movie database.

Overview refers to the description of a certain movie, and Tagline is the equivalent to the Summary in the Re:Voire database. Cast contains the director, editor and other members of each movie, but only the Director will be used further on, since he is the equivalent to the to the Author. The other features are considered to be the same.

In [26]:

```
md = md.drop(['budget','homepage','vote_count','original_language','original_title','po
pularity','production_companies','production_countries','release_date','revenue','runti
me','status','vote_average','spoken_languages'],axis=1)
md.shape
```

Out[26]:

```
(4803, 8)
```

The new data-frame contains 4803 movies with now 8 columns of information, which will be used as feature vectors for our Recommender system.

The recommender system

The first step for building the recommender system is to build an engine that computes the similarity between movies based on certain criteria. It should then suggest movies that are similar to the active movie which the user is currently watching. The first instance of the the Recommender system uses the movie Overview and Taglines (from the content wrangled above) to produce the prediction. This will be judged qualitatively, with the use of the TF-IDF technique, since there is no quantitative metric in the dataset. SciKit-learn contains a tool for this technique which eases the whole process.

In [27]:

```
md['tagline'] = md['tagline'].fillna('')
md['description'] = md['overview'] + md['tagline']
md['description'] = md['description'].fillna('')
```

In [28]:

```
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(md['description'])
tfidf_matrix.shape
```

Out[28]:

```
(4803, 110167)
```

Cosine Similarity

After the TF-IDF technique has produced the term weight vectors, the system needs a similarity measure for describing the proximity of two weighted vectors. The Cosine similarity measure will produce a numeric quantity that denotes the similarity between two movies. Also, after calculating the dot product from the TF-IDF Vectorizer function we will have a cosine similarity score. The Linear_kernel function from Sklearn will produce a much faster result.

In [29]:

```
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim[0]
```

Out[29]:

```
array([1.        , 0.00528077, 0.        , ..., 0.        , 0.        ,
       0.        ])
```

The code above has produced a cosine similarity matrix for all movies. Next, the system needs a function that extracts the current information from the similarity measure and produces a list with the most similar movies based on the score.

In [30]:

```
md = md.reset_index()
titles = md['title']
indices = pd.Series(md.index, index=md['title'])

def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    movie_indices = [i[0] for i in sim_scores]
    return titles.iloc[movie_indices]
```

Now the function can be used to generate the recommendations. As stated the system will work based on the current movie being watched. For simplicity the system uses the title, rather than the id of movies, as this is easier to test and understand. The calling of the function bellow with the name of the movie "The Dark Knight" will give the 10 most relevant movies according to the Overview and Tagline of the movies.

In [31]:

```
get_recommendations('The Dark Knight').head(10)
```

Out[31]:

```
3                         The Dark Knight Rises
290                             Batman Forever
412                             Batman Returns
2990    Batman: The Dark Knight Returns, Part 2
1251                                    Batman
1102                                       JFK
117                              Batman Begins
827                        Law Abiding Citizen
198          Sherlock Holmes: A Game of Shadows
9            Batman v Superman: Dawn of Justice
Name: title, dtype: object
```

Continued system

So far the system generates good predictions based on the movie description that has been used. This prediction engine is equivalent to the one the Re:Voire system will use to generate its movie recommendations with its experimental movies. But it is still incomplete, since it doesn't use the other features of the data-set. The following code will generate predictions using the Keywords, Author and Genres.

In [35]:

```
md = md[md['id'].isin(links_small)]
md.head(1)
```

Out[35]:

	index	genres	id	keywords	overview	tagline	title	cast	
0	0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	In the 22nd century, a paraplegic Marine is di...	Enter the World of Pandora.	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_ "52fe48("de...

In [36]:

```
md['keywords'] = md['keywords'].apply(literal_eval)
md['crew'] = md['crew'].apply(literal_eval)
md['crew_size'] = md['crew'].apply(lambda x: len(x))
md['genres'] = md['genres'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
```

In [37]:

```
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

In [38]:

```
md['director'] = md['crew'].apply(get_director)
md['director'] = md['director'].astype('str').apply(lambda x: str.lower(x.replace(" ", "")))
md['director'] = md['director'].apply(lambda x: [x,x, x])
```

The system needs to strips spaces and convert the characters into lower cases for the director's name, otherwise the engine will consider Steven Spielberg and Steven Soderbergh as the same person, confusing the algorithm.

In [39]:

```
md['keywords'] = md['keywords'].apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])

s = md.apply(lambda x: pd.Series(x['keywords']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'keyword'
s = s.value_counts()
s[:5]
s = s[s > 1]
```

The keywords feature, needs to be preprocessed before it can be used. We can see the frequency counts of the keywords that are included in the dataset, with the least instances occurring once and the most frequent 324 times. In a recommender system, words that occur only once are not useful for recommendations. Also, some words are such as Bird and Birds are considered with the same meaning, the only difference is that one is plural. The system cannot distinguish this so it needs to consider them the same. The code bellow works out this with no difficulty.

In [40]:

```
def filter_keywords(x):
    words = []
    for i in x:
        if i in s:
            words.append(i)
    return words

md['keywords'] = md['keywords'].apply(filter_keywords)
md['keywords'] = md['keywords'].apply(lambda x: [SnowballStemmer('english').stem(i) for
 i in x])
md['keywords'] = md['keywords'].apply(lambda x: [str.lower(i.replace(" ", "")) for i in
 x])
```

The next step is to generate the recommendations. This will be done a little different than the previous approach. The engine will put all of the features for every movie in one metadump. The Count Vectoriser tool from Scikit-learn will then create a count matrix for this metadump, like the one used before with the TF-IDF technique. After that the Cosine Similarity measure will help generate the most similar movies recommendation of a current movie.

In [41]:

```
md['metadump'] = md['keywords'] + md['director'] + md['genres']
md['metadump'] = md['metadump'].apply(lambda x: ' '.join(x))

count = CountVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='englis
h')
count_matrix = count.fit_transform(md['metadump'])
cosine_sim = cosine_similarity(count_matrix, count_matrix)

md = md.reset_index()
titles = md['title']
indices = pd.Series(md.index, index=md['title'])
```

Now the engine is ready to generate recommendations. The function initiated earlier can be reused here.

In [42]:

```
get_recommendations('The Dark Knight').head(10)
```

Out[42]:

```
3                    The Dark Knight Rises
117                          Batman Begins
1117                          The Prestige
95                               Inception
963                               Insomnia
2838                               Memento
94                             Interstellar
412                           Batman Returns
203                           Batman & Robin
9       Batman v Superman: Dawn of Justice
Name: title, dtype: object
```

The recommendations here are different than the previous ones, and possibly a little more accurate, because of the different data used. The Author has made a great impact on the recommendations as Christopher Nolan's movies are more dominant in this situation. In the field of Experimental cinema, the Author and his touch is one of the most important aspect people expect to see when viewing experimental cinema, so this is taken into consideration in the example.

This recommendation system only considers movies as a means to suggest items. In short, the system doesn't capture personal opinions and biases, so anyone who watches movies will have the same suggestions. This is a major flaw for the ultimate goal. Most systems utilise the Users and their ratings, hence the user input needs to be included. By using this information, the system will be capable of capturing the tastes of the users and generate personal recommendations for each individual user.

Collaborative Filtering

As mentioned earlier in the project, the Collaborative filtering approach is based on the idea that similar users can be used to predict if a movie is going to be liked by another similar user who has never seen the item. Hence the recommendation will predict ratings based on how other users have rated the movie. The data required for this is taken from the MovieLens dataset of users that contains 100,000 ratings from 700 users on 9000 movies. Assuming that Re:Voire will use a similar database for users who rate movies the same way the MovieLens users have rated popular feature movies.

This approach will be implemented first as a stand alone feature and after that, it will be combined with the previous engine to create a hybrid recommendation system. The Collaborative Filtering will be handled with the powerful Singular Value Decomposition from the Surprise library, in order to minimise the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE).

In [43]:

```
users = pd.read_csv('../ratings.csv')
users.head()
```

Out[43]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

The ratings subset from the MovieLens dataset contains the IDs of each user, the movieID for each movie rated by the user and the ratings given from the scale of 1 to 5. First the data will be divided using the 5 fold cross validation model. This will ensure that the SVD model will generate a more precise prediction further on.

In [45]:

```
reader = Reader()

data = Dataset.load_from_df(users[['userId', 'movieId', 'rating']], reader)
data.split(n_folds=5)

svd = SVD()
evaluate(svd, data, measures=['RMSE', 'MAE'])
```

```
Evaluating RMSE, MAE of algorithm SVD.

------------
Fold 1
RMSE: 0.8915
MAE:  0.6873
------------
Fold 2
RMSE: 0.8940
MAE:  0.6904
------------
Fold 3
RMSE: 0.9043
MAE:  0.6947
------------
Fold 4
RMSE: 0.9103
MAE:  0.7018
------------
Fold 5
RMSE: 0.8815
MAE:  0.6778
------------
------------
Mean RMSE: 0.8963
Mean MAE : 0.6904
------------
------------
```

Out[45]:

```
CaseInsensitiveDefaultDict(list,
                           {'mae': [0.6873407587584036,
                             0.6903974214749422,
                             0.6946682560033739,
                             0.7018313564847312,
                             0.6777902298460609],
                            'rmse': [0.8915151487602889,
                             0.8939982755729916,
                             0.9042682917587916,
                             0.9103344746735726,
                             0.8814669996378004]})
```

The RMSE is 0.89

After we have optimised the SVD algorythm, the recommender system needs to make a training set in order for the pridictions to be generated. The training set contains the decomposed values.

In [46]:

```
trainset = data.build_full_trainset()
svd.train(trainset)
```

Out[46]:

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f3be69a4ef0
>
```

Now the Surprise toolset can generate predictions based on the training data that was injected. The prediction model works by picking a user who has already rated some movies, and picking a random movie. The system takes this as input and will generate a prediction on the score based on the estimates in the training set. The score is based on the range between 1 to 5 as in the movie ratings dataset In simple terms, the score determines weather the user will like or dislike the movie.

In [47]:

```
svd.predict(1, 55, 3)
```

Out[47]:

```
Prediction(uid=1, iid=55, r_ui=3, est=2.461484480114258, details={'was_imp
ossible': False})
```

In this function 1 is associated with the user ID and 55 corresponds to the movie ID. The result is given from the SVD predictor and can be interpreted as, user 1 has rated movie 55 with an estimated score of 2.72. In other words, the system predicts that the user will rate movie 55 with this score.

Hybrid System

As mentioned before this Collaborative filtering model as it stands, works only to estimate scores. It doesn't take into consideration what the movie is, or what it contains. This is definitely not enough for a proper movie recommendation system to work. Combining both methods together to create a hybrid system is essential. This hybrid system will use the aspects that what we have developed so far to generate personalised recommendations.

If we consider this to be the Re:Voire system, it will be able to recommend experimental films which it is estimated that individual users will like, based on the information given in the context of the movie and the other users ratings. The system will take UserID and Movie Title as inputs and will output Similar movies sorted on the basis of expected ratings.

In [72]:

```
def convert_int(x):
    try:
        return int(x)
    except:
        return np.nan

md.head(1).transpose()
```

Out[72]:

	0
level_0	0
index	0
genres	[Action, Adventure, Fantasy, Science Fiction]
id	19995
keywords	[cultureclash, futur, spacewar, spacecoloni, s...
overview	In the 22nd century, a paraplegic Marine is di...
tagline	Enter the World of Pandora.
title	Avatar
cast	[{"cast_id": 242, "character": "Jake Sully", "...
crew	[{'credit_id': '52fe48009251416c750aca23', 'de...
description	In the 22nd century, a paraplegic Marine is di...
crew_size	153
director	[jamescameron, jamescameron, jamescameron]
metadump	cultureclash futur spacewar spacecoloni societ...

In [50]:

```
id_map = pd.read_csv('../links.csv')[['movieId', 'tmdbId']]
id_map['tmdbId'] = id_map['tmdbId'].apply(convert_int)
id_map.columns = ['movieId', 'id']
id_map = id_map.merge(md[['title', 'id']], on='id').set_index('title')
#id_map = id_map.set_index('tmdbId')
```

In [51]:

```
indices_map = id_map.set_index('id')
```

In [52]:

```
def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    #print(idx)
    movie_id = id_map.loc[title]['movieId']

    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]

    movies = md.iloc[movie_indices][['title', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]
['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

The system is now all set up to generate predictions based on User input (ratings) and the context of movies. The recommendation for each user consists of 10 movies compiled from the content and grouped by their estimated ratings in a descending order. The following example shows the predictions for two people who are watching the same movie, but have slightly different recommended movies based on the predicted score they are assumed to give. After this the system will be tested on real users and their results will be considered as a validation to the system's ability to generate predictions.

In [53]:

```
hybrid(1, 'Avatar')
```

Out[53]:

	title	id	est
270	Terminator 2: Judgment Day	280	3.254327
2747	The Terminator	218	3.112695
2063	Aliens	679	3.094833
46	X-Men: Days of Future Past	127585	3.094260
47	Star Trek Into Darkness	54138	2.879391
2011	Predator	106	2.825475
819	Superman II	8536	2.733227
1904	The Covenant	9954	2.725979
4	John Carter	49529	2.679809
72	Suicide Squad	297761	2.629352

In [74]:

```
hybrid (7, 'Avatar')
```

Out[74]:

	title	id	est
2747	The Terminator	218	3.984383
2063	Aliens	679	3.737076
47	Star Trek Into Darkness	54138	3.552689
270	Terminator 2: Judgment Day	280	3.543133
2011	Predator	106	3.500529
556	The Abyss	2756	3.498759
1904	The Covenant	9954	3.357531
46	X-Men: Days of Future Past	127585	3.356254
510	Titan A.E.	7450	3.275952
251	Ender's Game	80274	3.251230