University of Applied Sciences Cologne

Master's Thesis

Recommender Systems for Live-Transmission in interactive Television: Gathering and Rating User Feedback

Master of Science in Media Informatics

Faculty of Computer Science and Engineering Science

Martin Gude, B.Sc.

Supervisors: 1. Prof. Dr. Stefan Grünvogel

2. Prof. Dr. Gerhard Plaßmann

Cologne, November 2007

Abstract

Recommender systems have been strongly researched within the last decade. They have strongly progressed over the time for non-live environments. However, they do not meet the requirements of live broadcast. Live broadcast suffers from several inherent problems, e.g. the impossibility to foresee the contents of the next items or the reactions of the user to the changing programme.

In this thesis a simple model of the complex live broadcast environment is developed to classify the media assets and the reactions of the viewer. A Content-Based and a hybrid recommendation algorithm are tested. The simple but powerful Content-Based algorithm has proven to calculate high quality recommendations from low quality data. The hybrid approach does not perform that well. However, this is caused by the poor base data.

The thesis can serve as a basis for further live recommender system research.

Keywords:

Recommender systems, interactive TV, Live TV, sports transmission

Table of Contents

List of Abbreviations

CB	Content-Based Filtering
CBCF	Content-Boosted Collaborative Filtering
CF	Collaborative Filtering
EPG	Electronic Programme Guide
GUI	Graphical User Interface
IPTV	Internet Protocol Television
iTV	Interactive Television
MAE	Mean Absolute Error
ROC	Receiver Operating Characteristics
RS	Recommender System
SBD	Scenario-Based Development
UML	Unified Modelling Language

List of Symbols

a	Actual user
$c_{u,i}$	Content-Based prediction for user u and item i
$hm_{a,u}$	Harmonic mean weighting factor for user a and u
$hw_{a,u}$	Hybrid correlation weight for user a and u
i	Any item, also used as shorthand for item $i_{j,k}$ in channel j at position k
$\mathfrak{I}$	Set of all available items
$I_{u,t}$	Set of all items tagged with t and rated by user u
$p_{a,i}$	Prediction calculated for user a and item i
$P_{a,u}$	Pearson Correlation between the user a and u
$r_{u,i}$	Rating user u gave to item i
$sg_{a,u}$	Significance weighting factor for user a and u
sw_a	Self-weighting factor for user a
t	Tag

$\mathfrak{T}$	Set of all tags associated with any item
u	Any user
$U_{u,t}$	User profile for user u and tag t
V	Pseudo user rating Matrix containing all v_u
$v_{u,i}$	Pseudo user rating for user u and item i
tw_u	Time weighting factor for user a and u
$l_{\min}$	Minimal number of training items to deliver good predictions

List of Figures

Preface

This master's thesis has been written as part of the European research project "LIVE – staging of Media Events" at Pixelpark AG, Cologne. Its results are used and published in work package 6 of that project.

The LIVE project aims to create a whole new user experience for television. It attempts to improve the classical linear approach of broadcasting live sport events. It creates new formats and services and enables the production of new non-linear multi-stream shows.

Recommender systems form an important basis for interactive television. Within the LIVE project, they are to provide the consumer with personalised TV streams and the producer with recommendations for programme creation.

A CD is bundled with this thesis. It contains the thesis in PDF format, the test application, the data gathered in the user tests and the python library elementree 1.2.6 used in the test application.

1 Introduction

Recommender systems are a means of personalisation providing their users with personalised recommendations of items that would possibly suit the users' needs. They are used in broad area contexts where items are somehow linked to users. Recommender systems have been thoroughly researched within the last decade.

Recently recommender system research emerged from time independent items like books or movies to an automatic generation of personalised streams. Up to now, these streams reside mainly in the music area but they arise also in fields like interactive TV.

This thesis proposes an algorithm for building personalised streams within interactive live TV. The development of the algorithm comprises a basic model for users and media items.

As the character of live TV leads to problems for recommender systems, some means have to be introduced to overcome those problems. The biggest problem for live TV is that you can never foresee the exact contents of an item, so that some substitute data has to be introduced.

The algorithm is developed with possible usage scenarios in mind, in order to reduce the required interaction effort. It is created in order to build high quality recommendations out of low quality data.

2 General Overview about Recommender Systems

Recommender Systems play an important role within interactive Media. Once a registered user performs an action, it is recorded and a specific profile is being built up. Based upon this profile, the recommender system forms personalised recommendations for that user by predicting how that user would like, i.e. rate, that item. The recommender system "typically recommends items with high predicted ratings" (Cosley et al., 2003).

Usually recommender systems build upon collaborative filtering algorithms, i.e. the system recommends one certain item by matching the user's profile with profiles of other users. Then the system recommends items similar users liked afore. Depending on whether an item can be used several times, an item will also be recommended several times.

One other type of recommender systems uses Content-Based methods that semantically analyse the items. These representations of the items in the user profile are subsequently compared to possibly recommended items. If they matched with each other, the tested item would be recommended to the user.

2.1 Where Recommender Systems are used

Recommender systems are used in a wide variety of contexts. These contexts range from online shops to personalised media streams. Recommender systems can be found in classical non-mobile appliances and lately also in mobile devices.

In general you could say that a recommender system can be used in every context where a user is to choose among several different options. The system assists the user to select one good-fitting item. If collaborative filtering was used, the type of those items would not be important as long as you can save some metadata for them.

2.2 Taxonomies: Folksonomy and Formal Terminologies

The term taxonomy refers to a “classification of things” and its underlying principles. Taxonomic schemes can be used to classify almost anything (Wikipedia, 2007a). Jacob (Jacob, 2004) distinguishes between “classification” and “categorization” which will be discussed later. Taxonomies are used to describe items for later retrieval. In the case of recommender systems, taxonomies are used to create metadata about media assets.

Classical taxonomic schemes in the field of information technology are driven by pre-optimised formal terminologies, i.e. taxonomies using a controlled vocabulary, created by domain experts, e.g. Semantic Web ontologies (Halpin, Robu & Shepherd, 2007). These taxonomies, referred by Jacob as classifications, require “systematic arrangement of classes of entities based on analysis of the set of individually necessary and jointly sufficient characteristics that defines each class.” In a classification system “classes are mutually exclusive and non-overlapping” so that the adjacent boundaries are completely fixed. Furthermore “classification involves a process of identification (or definition) in that it asserts a meaningful, one-for-one relationship between an entity and its class.” Consequently, an entity can either be a class-member or not. The assignment to one class is rigorously “predetermined by guidelines or principles” (Jacob, 2004).

Formal terminologies can describe the items relatively well but suffer from the vocabulary problem (Marlow et al., 2006): people tend to use a “great variety of words to refer to the same thing.” Using formal terminologies requires the system users to learn the specific classification terms. Those terms never really fit the user needs as “there is no one good access term for most objects.” (Furnas et al., 1987) Disagree-

ment on the terms of resource description “can lead to missed information or inefficient user interaction” (Marlow et al., 2006).

Recently a new approach has emerged within the space of naming resources: Tagging. This approach “allows users to ‘tag’ objects with keywords”, i.e. classifying or, according to Jacob, categorizing resources with loosely associated keywords. Compared to classical terminology-driven approaches, tagging is much more flexible and efficient as adding loose tags to a resource is much simpler than deciding to which degree it fits into a predefined category. Usage of tag synonyms results, as a trade-off, in “informational redundancy” but this can help to overcome the aforementioned vocabulary problem. Tagging taxonomies are not less stable than terminologies though there is no controlled vocabulary. “Stable” means that users have “some consensus” which keywords best describe an item and that those keywords are used in most cases. (Halpin, Robu & Shepherd, 2007; Marlow et al., 2006)

Halpin et al. (Halpin, Robu & Shepherd, 2007) describe tags as links between users and resources. They identify a “tripartite structure of tagging”. According to them, tagging systems consist of the user space, the tag space and the resource space. Each space is a set of all specific items. The “tags provide the link between the users of the system and the resources.”

Social tagging systems allow the users to share the tags amongst them (Marlow et al., 2006). As a result tagging taxonomies are, in contrast to formal terminologies, user-created and user-maintained. They have recently been termed folksonomies as a neologism of folk and taxonomy. That neologism does not refer to folk taxonomies as documented in anthropological work but to that new approach of shifting the control of classification vocabulary to the user. (Wikipedia, 2007b)

2.3 Content-Based Filtering

Content-Based recommender systems recommend items “by comparing representations of content contained in an item to representations of content that interests the user.” In most cases, a keyword profile is built. Apparently, this works well in text

domains but not in domains where there is not much content associated with the items or where a computer cannot easily analyse this content. Relying on rich descriptions, Content-Based recommender systems need significant knowledge-engineering efforts to create substantial metadata for the items. However, Content-Based filtering can uniquely characterize the user. (Melville, Mooney & Nagarajan, 2001; Cosley et al., 2003; O'Donovan & Smyth, 2005)

Content-Based systems "form profiles for each user independently." Even if two items were in two neighbour categories that people normally like both (e.g. fantasy and science fiction) the item from category A would never be recommended to the user if he only rated items from category B.

Taxonomies can be used as one possible means to add metadata to content items. Which of the aforementioned taxonomies suits the needs better is highly dependant on the context it should be used in.

2.4 Collaborative Filtering

Collaborative filtering systems compute profile similarity between the other users and the target user "by comparing users' opinions of items." Profile similarity is usually computed by comparing rating-vectors with various distance metrics, e.g. Pearson correlation or cosine similarity. They provide the user with the "best bets" he will most likely be interested in, either one single item or a "ranked list of items" – usually referred as *top-N-items* with *N* between 1 and 20. (Cosley et al., 2003; McLaughlin & Herlocker, 2004)

In contrast to Content-Based systems, recommender systems based on collaborative filtering can provide the user with unexpected but fitting recommendations that do not have anything in common with aforerated items. Collaborative filtering is a very successful methodology in almost every domain – especially "where multi-value ratings are available". (McLaughlin & Herlocker, 2004)

However they suffer from two key problems: sparsity and First-rater problem. As most users only rate a small portion of all items, it is highly difficult to find users

with “significantly similar ratings.” Furthermore an item cannot be recommended before one user has rated it. This can be the case if the item has newly been introduced to the system or if it is a rather obscure niche item. (Melville, Mooney & Nagarajan, 2001)

Collaborative filtering systems can be implemented using neighbourhood-based algorithms. Melville et al. (Melville, Mooney & Nagarajan, 2001) introduce a possible implementation of a neighbourhood-based algorithm using a Pearson correlation coefficient (1) to measure similarity between two users.

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)^2 \sum_{i=1}^{m}(r_{u,i} - \bar{r}_u)^2}} \quad (1)$$

In this equation $r_{a,i}$ is the rating given to item i by user a and $\bar{r}_a$ is the mean rating user a gave overall.

In a second step n users are selected, that have the highest profile similarity with user a. These users constitute the neighbourhood of a.

In step 3, predictions for all items are computed “as the weighted average of deviations from the neighbours mean.” In equation (2) $p_{a,i}$ is the prediction for user a and item i.

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u)P_{a,u}}{\sum_{u=1}^{n} P_{a,u}} \quad (2)$$

Collaborative filtering would not perform well, if a high correlation with his neighbours was based on very few co-rated (i.e. overlapping) items. To eliminate those bad predictors the correlation is multiplied with a significance weighting factor $sg_{a,u}$ (3).

$$sg_{a,u} = \begin{cases} \frac{n_{a,u}}{50} & \text{if } n_{a,u} < 50 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

In this equation $n_{a,u}$ is the number of items user *a* and user *u* have co-rated.

2.5 Content-Boosted Collaborative Filtering

Melville et al. (Melville, Mooney & Nagarajan, 2001; Melville, Mooney & Nagarajan, 2002) developed an approach to overcome the problems with collaborative filtering and Content-Based filtering which are mentioned before. They use Content-Based predictions to "convert a sparse user ratings matrix into a full user ratings matrix." Then they apply collaborative filtering on this matrix to compute recommendations. Their approach, Content-Boosted Collaborative Filtering (CBCF), performs significantly better than recommender systems using either Content-Based or collaborative filtering algorithms.

The first step for CBCF is to create a pseudo user ratings vector v_u for every user *u*. The element $v_{u,i}$ of this vector is either the item rating by that user ($r_{u,i}$) or the rating prediction computed by a Content-Based predictor ($c_{u,i}$) if the user had not rated the item before (4). All vectors v_u form the pseudo ratings matrix *V*.

$$v_{u,i} = \begin{cases} r_{u,i} & \text{if user } u \text{ rated item } i \\ c_{u,i} & \text{otherwise} \end{cases} \quad (4)$$

As the accuracy of the Content-Based prediction depends on the number of items the user has already rated, a harmonic mean weighting factor ($hm_{a,u}$) is introduced (5). If a user had rated many items, the Content-Based predictions would be good and the resulting pseudo user-ratings vector is accurate. In contrast, that vector will not be accurate if the user only rates a few items. The inexactness in that vector leads to high correlations between the active user and the other users even though they should not be that highly correlated. In order to give exacter predictions, the correla-

tions between two users are weighted with this factor. In this equation n_u refers to the number of items user u has rated.

$$hm_{a,u} = \frac{2m_a m_u}{m_a + m_u}$$
$$m_u = \begin{cases} \frac{n_u}{50} & \text{if } n_u < 50 \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

The harmonic mean biases the weight to the lower value (m_a and m_u). Correlations between two pseudo user-ratings with at least 50 ratings each will receive the highest weight, whereas correlations with less rated items will be devalued. The level of 50 rated items was empirically chosen based on the learning curve of the used content predictor.

To the harmonic mean weight, the significance weighting factor (3) is added to form the hybrid correlation weight $hw_{a,u}$ (6).

$$hw_{a,u} = hm_{a,u} + sg_{a,u} \quad (6)$$

Collaborative filtering algorithms compute a prediction for one user "as a weighted sum of the mean-centered votes of the best-n neighbors of that user." In CBCF, the pseudo active user, i.e. the pseudo user-ratings vector for that user, is incorporated into the neighbourhood. To raise confidence in the Content-Based predictions a self weighting factor sw_a (7) is used:

$$sw_a = \begin{cases} \frac{n_a}{50}\max & \text{if } n_a < 50 \\ \max & \text{otherwise} \end{cases} \quad (7)$$

The parameter n_a is the number of items rated by user a. As for equation (6), the threshold 50 is chosen in accordance to the learning curve of the Content-Based predictor. The parameter *max* indicates the confidence in the Content-Based predictor. In their experiments, Melville et al. used a value of 2.

The final CBCF prediction $p_{a,i}$ for the active user a and item i is a combination of both weighting schemes (8).

$$p_{a,i} = \bar{v}_a + \frac{sw_a(c_a - \bar{v}_a) + \sum_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u} P_{a,u} (v_{u,i} - \bar{v}_u)}{sw_a + \sum_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u} P_{a,u}} \quad (8)$$

In the CBCF prediction equation $c_{a,i}$ is the pure content prediction for user a and item i. $v_{u,i}$ is the pseudo user-rating calculated for user u and item i. $\bar{v}_a$ and $\bar{v}_u$ are the mean rating over all items of user u's respectively user a's pseudo rating vector, n is the number of neighbours.

2.6 Comparison of the Filtering Approaches

Compared with a pure-Content-Based predictor, a pure collaborative filtering predictor and naïve hybrid approach, content-boosted collaborative filtering always performs significantly better. Even though, compared to the pure CF-approach, CBCF does not perform much better. This was tested using two different metrics, viz Receiver Operating Characteristic (ROC) and mean absolute error (MAE).

With CBCF it is more likely to find a better neighbourhood than with pure collaborative filtering. Users can be considered as neighbours even though they have not co-rated any item but are considered to have a similar taste through Content-Based predictions.

As calculating the pseudo user ratings matrix is highly complex, computational costs need to be reduced. One possibility is storing a complete pseudo ratings matrix and performing only incremental updates when they are needed, e.g. if user (re-)rates an item.

As both components of the CBCF system, i.e. the Content-Based predictor and the collaborative filtering algorithm, are independent, the prediction can be improved by improving the basic components.

Taking the computational overload into account, it has to be considered whether CBCF affords benefits over pure collaborative filtering.

2.7 Examples of Recommender Systems

In the following part, three well established commercial recommender systems, Amazon, del.icio.us and last.fm, are briefly examined. In addition a fourth recommender system, Lifetrak is described in more detail. Lifetrak is developed at the University of California, Los Angeles.

All commercial recommender systems set another focus of recommendation. Amazon provides classical time independent recommendation using collaborative filtering. Del.icio.us uses some hybrid CF and tag-based, i.e. content-based, algorithm. Last.fm's recommender system is also based on collaborative filtering but also incorporates tags into its algorithm. All these recommender systems do not provide much information about how they work in detail. Most of the description is accordingly inferred from FAQs as well as third party sources.

Lifetrak is, in contrast to the other systems, an academic project which has been well documented. At the time of writing, it cannot be obtained for usage yet. Lifetrak is designed to operate on a mobile device. It uses a Content-Based predictor. It takes into account several environmental factors for its recommendations.

2.7.1 Amazon

The media store Amazon[1] is one of the biggest e-commerce sites in the entire world. Amazon sells goods like books, CDs or DVDs. Recently, a bunch of other product

[1] http://www.amazon.com

categories such as “consumer electronics” or “food and household” have been added. Nowadays, Amazon offers products for nearly all aspects of everyday life.

Amazon has one of the best-known recommender systems that are currently in use. It usually serves as an example for non-experts of what a recommender system is like. Amazon’s recommender system uses collaborative filtering based on the items a customer has purchased before. Furthermore, the user can rate any item on a graphic 5-star rating scale.

Amazon also recommends items that are similar to the item the customer currently looks at, e.g. “Better Together” (popular combination of two items) or “Customers Who Bought This Item Also Bought”. Amazon uses the recommender system to promote cross-selling on the customer purchases. (Amazon.com, 2007)

2.7.2 del.icio.us

The social bookmarking application del.icio.us[2] uses a recommender system to display currently “hot” websites on its homepage. It uses tags to describe the semantics of the content. Consequently, a folksonomy classifying all items is built up. Related tags can be computed by using the folksonomy as a thesaurus.

Del.icio.us is commonly used as a research tool, but it can also be seen as some kind of trend barometer (Kleske, 2006). The del.icio.us recommendation engine takes tags associated with minimum 10 items to build customised recommendations (Schachter, 2005). On the homepage, as of August 2007, a hotlist containing currently popular items is displayed. In the right column, tags and adherent items that might be interesting for the user are recommended.

[2] http://del.icio.us

2.7.3 last.fm

Last.fm[3] is a social music platform offering personalised radio streams to its users. It uses the recommender system Audioscrobbler[4] to build personalised recommendations. Audioscrobbler builds a profile of music tracks the user listens to. The user profile can either be populated ("scrobbling") by listening to the last.fm internet radio service or by listening to one's personal music collection. Additionally, the user can rate any single track on a dual scale ("Love it" or "Hate it"). If the user does not vote he implicitly votes however. As the listening time is also taken into account for the recommendations, not voting can be described as a neutral-positive opinion about that track. Audioscrobbler uses a collaborative filtering algorithm for its recommendations.

Listening to a personalised radio stream is possible with the last.fm player. It is available as flash and stand-alone version for all important platforms (Windows, Mac OS and Linux). Installing the stand-alone version also installs plug-ins into the users' music players, e.g. iTunes, Winamp or QuodLibet. Using these plug-ins, the user can also scrobble tracks played with that media player. Audioscrobbler is able to scrobble tracks played on portable players such as the Apple iPod, Microsoft Zune or Creative's Zen players.

Last.fm also incorporates tagging functions into its service to generate metadata for the tracks. Tags are usually used to describe genre, mood or artist characteristics, but any other classifications are used, too.

Last.fm offers several personalised radio streams: My Recommendations, My Neighbourhood, My Radio Station and My Loved Tracks. The two last-mentioned are only available to paying users (subscribers). In addition two variations of tag radios are available: the overall tag radio and a user tag radio (subscribers only). Both radios only play songs tagged with certain tags, the user tag radio plays only songs tagged by one user. (Wikipedia, 2007c)

---

[3] http://www.last.fm

[4] http://www.audioscrobbler.net

2.7.4 Lifetrak

Reddy and Mascia (Reddy & Mascia, 2006) developed the music player Lifetrak that aims to provide the listener with context-aware music experience. Its goal "is to liberate the user from having to consciously specify the music that they want to play." Lifetrak is designed to work on a mobile device, viz the Nokia 770 Internet Tablet. It uses a context-sensitive music-engine to decide which music to play. It takes four factors into account: the location of the user, the current time, the velocity of the user and the user's urban environment, i.e. the sound level of the environment, the local traffic conditions and the current weather. The context engine is adjusted by a learning model based on user feedback.

Lifetrak does not analyze the songs and connect them automatically to the different contexts as not all users are likely to hear the same music in the same situation. Lifetrak requires the user to tag the song database with the context constructs. [5] Then the system evaluates the current context and builds a situation-specific playlist.

Lifetrak consists of four main components. The user space document contains a list of all songs and the associated contexts. The context engine builds context information from several sensors. It stores the information in tags: if a tag is associated with the context it has the value 1 otherwise 0 at the beginning. The rating generator creates a ranked playlist from user context and song database. The music player provides an interface to adjust the current settings.

The song rating algorithm is rather simple. It uses some kind of Content-Based algorithm multiplying the tag values of the songs with the values the context engine provided. Then all tag values are summed up. This rating is to represent the desire of the user to listen to a certain song.

[5] Reddy and Mascia do not use the term tag as it has been previously defined. They use tags more in a classical terminology driven way as they provide the user with a distinct predefined vocabulary.

User feedback is also realised in a rather simple way. The first method is using a context equalizer that weights the current context vector with its appropriate weighting factor. Secondly, a dual “love it” and “hate it” feedback mechanism is used. For each tag coming from the current context, the value is increased respectively decreased.

2.7.5 In a Nutshell

All recommender systems examined above perform well in their respective area. However, not all of those approaches would perform well in interactive TV, especially in live TV. Amazon and del.icio.us do not suit the requirements for live media but Lifetrak and especially last.fm provide some interesting features.

Last.fm’s model for gathering ratings by taking usage time as some kind of positive feedback plus explicit rating one specific song enables the user to give that feedback very easily.

Lifetrak’s context-aware approach is also very interesting even though it might be more difficult to obtain external data. Even though, some kind of mood information should be taken into account.

3 Recommender Systems in the Context of Live-Transmissions

Live transmission evokes special problems for recommender systems. In contrast to classical recommender systems, it is impossible to predict the content of any media asset. Consequently, any Content-Based prediction can only be a reaction to an event that happened in the closer past. Collaborative filtering relies on enough consumers watching the programmes bouquet.

In this chapter, ideas of tasks a recommender system could perform and the problems it would have to cope with are envisioned. Furthermore, differences between lean-back and lean-forward scenarios are briefly described.

3.1 Tasks of a Recommender System in Live Media

In contrast to archive-based media, a recommender system in the context of live transmission is faced with several challenges.

Live transmission induces two types of recommender systems: a basic implementation similar to an electronic programme guide (EPG) and a more sophisticated implementation examining especially the preceding period of time.

A personalised EPG can provide the consumer with long term information which broadcast could be interesting for him. As a timetable for events is usually available this is an excellent foundation for any recommender system in live transmission.

A personalised EPG enables the user to discover when to best watch TV for the most thrilling experience. The user could build anything like a playlist respectively a reminder list for the upcoming broadcasts.

In addition, a recommender system for live TV must go much further. It does not suffice to measure the user and the upcoming programme once. Rating and measuring has to be an ongoing process. A recommender system should enable the consumer to watch the programme he wants to watch no matter what he specified afore. In contrast, the system should be able to adjust and weighten recommendations according to recently changed parameters.

3.2 Lean-back vs. Lean-forward

Recommender systems can provide two different user experiences to the viewer: lean-back and lean-forward. In a lean-back scenario, the viewer rather passively consumes the TV programme whereas in a lean-forward approach the viewer actively interacts with the programme.

A lean-back approach would liberate the user from having to decide consciously which programme to watch. This usage scenario requires some conditions to be specified under which circumstances a recommendation should be followed.

In contrast, lean-forward would require the user to consciously decide whether to follow a recommendation or not. This induces an active participation of the viewer but the system does not patronize the user.

A recommender system for TV should afford both approaches. It should also be adjustable, which approach to use as the same viewer might once be interested in one approach and at another time in the other variant.

3.3 Problems for a Recommender System

Recommender systems supporting live transmission have to face some key problems due to the nature of live events. It is rather difficult to gather data with sufficient quality for both recommendation approaches, i.e. collaborative filtering and Content-Based filtering.

Content-Based filtering suffers from the impossibility of providing appropriate content descriptions. As you cannot predict what will happen next, you can only provide raw descriptions afore. Tagging a content-item appropriately is merely possible as a reaction to the closer past. A really exact description can only be assigned afterwards.

For example the viewer watches a football game but also wants to view all goals of a parallel game. When a goal is scored, the viewer can be shown a repetition of the goal, but it is impossible to show the goal in real time. Also showing chances for scoring a goal would be possible trade-off.

A crucial aspect for getting Content-Based filtering to work is providing the system with good annotations in real time. The media assets must be tagged by editors while being broadcast. The system can support the editor by providing a set of tags that could fit in the current context. Furthermore, it should be easy to introduce new tags to the system so that a folksonomy could be built.

Collaborative filtering relies on a critical mass of viewers that must be available. In contrast to non-real-time recommender systems it is impossible to take all registered viewers as a basis for collaborative filtering, but only the consumers currently watching the programme bouquet. Accordingly, you have to induce an action from imperfect data.

Consequently, it is necessary to use a combination of collaborative filtering and Content-Based filtering such as CBCF to overcome the problems both approaches suffer from.

4 Scenario-Based Development

Development of an interactive system requires an efficient development method. For this system, Scenario-Based Development (SBD) framework as introduced by Rosson & Carrol (Rosson & Carroll, 2002) is used.

SBD, as indicated by its name, uses scenarios to envision functionalities which are to be developed within the project. Due to their concrete and flexible character, scenarios can be easily altered and elaborated. They focus designers on the needs of the people in the real world.

SBD helps to manage trade-offs within the development process. Furthermore, it enables to integrate many different kinds of knowledge and facilitates participatory design by using a universal accessible language.

Scenario-Based Development integrates the advantages of a linear development process with iterative feedback and reworking. It resolves the trade-offs between waterfall models and flexible prototyping.

The SBD framework is segmented into three phases: analysis, design and prototyping and evaluation. Each stage has specified results that the next stage uses as input. The framework also enables iterative refinement of these results.

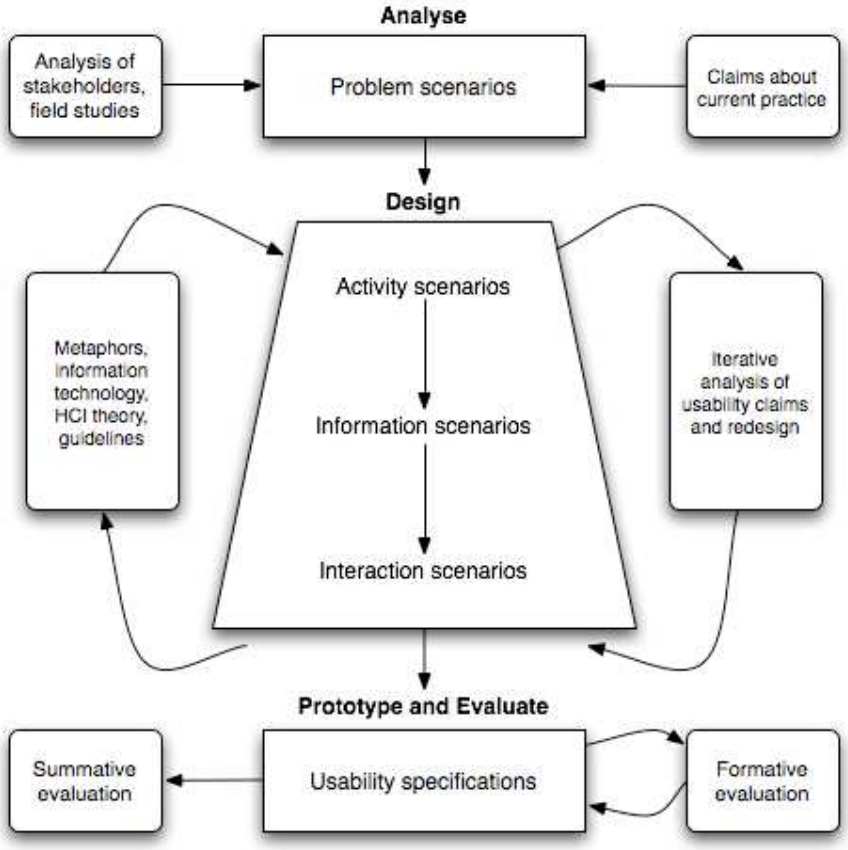

Fig. 1: Overview of the SBD framework (Rosson & Carroll, 2002)

Scenarios are transformed within the process from problem scenarios on the analysis stage via activity, information and interaction scenarios (design stage) to final usability specifications (prototyping and evaluation). Each stage results in claims about the particular scenarios. A claim describes the most important features including the pros and cons. It summarizes the impacts of a feature on the respective actor and on the other scenarios. Claims facilitate design reasoning by encouraging designers to increase positive and to decrease negative impacts.

Even though Fig. 1 looks to be a downward waterfall process from analysis to design and evaluation, it is intended to be an iterative process. Scenarios can be altered

and refined at any point during design. Accordingly, the adjacent scenarios have to be altered, too.

4.1 Analysis

SBD describes requirements analysis as an ongoing process throughout development, because it is impossible to specify all problems in advance. It aims to get involved all different stakeholders in the analysis phase as each task is described differently by each of them.

The first step in the analysis phase is to build a root concept that describes the general ideas of the project. In a second step, an analysis of current practice is carried out. Rosson & Carrol suggest arranging field studies that "examine the activities of various project stakeholders." Field studies can be undertaken as workplace observations, recordings, interviews or artefact analysis. The data gathered in field studies is summarised. These summaries serve as a basis for the problem scenarios and claims.

The resulting problem scenarios and the related claims can serve as requirements specification. They describe implicitly user needs and opportunities. Furthermore, problem scenarios also describe the characteristics of the users and their organisational context. Besides, descriptions of typical and critical tasks as well as the tools used by the stakeholders are presented.

4.2 Design

The design phase is divided in three parts: activity design, information design and interaction design.

Activity design specifies the system's functionality, i.e. what the system is to do. It aims to envision the entire situation including physical and social characteristics. Its computational perspective enhances the communication between usability experts

and developers. Activity scenarios lead to an analysis of trade-offs implied by new design features.

Information and interaction scenarios specify how a system performs the previously defined tasks. Information scenarios depict how information is dealt with and how the system presents it. Besides, they determine which information is shown to the user.

Interaction scenarios describe how the user should interact with the system. They describe concrete means of accessing and manipulating task information. The interaction design assures “that people can do the right things at the right time”.

4.3 Prototyping and Evaluation

Prototypes, concrete but partial implementations of system designs, “are constructed and evaluated to guide redesign and refinement.” They are used to investigate the problems occurring during system development. Several evaluation techniques are available to address the respective problems.

Prototyping and evaluation assure finding problems caused by the design ideas in an early stage. This process helps to iteratively develop new insights to the problem and thus reducing development costs. Hence, the design – and also the design process – ameliorate.

5 Scenario-Based Development of a Live Recommender System

In this chapter the results of the Scenario-Based Development of the live recommender system are described. Scenario-Based Development is used as method for requirements analysis. Hence, the process is not conducted until the end. The first three steps introduced by Rosson & Carrol – root concept, problem scenarios and claims and activity scenarios and claims – are used in this project. These steps serve as a basis for UML (Unified Modelling Language) Use Cases. Ensuring better usability, usually the key feature of Scenario-Based Development, is not the main goal of this project. In contrast, development of a recommendation algorithm is in the focus. However, SBD is a very good means to describe requirements.

5.1 Analysis

5.1.1 Root Concept

The root concept is derived from the overall project vision within the IST LIVE project (LIVE consortium, 2007). This part is mainly concerned with personalised content recommendations.

The recommender system builds a personalised live TV stream. This stream is broadcast to the consumer. The consumer is supplied with personalised recommendations he can follow or not. The aim of this part of the project is to develop an

algorithm that provides these personalised recommendations. In conclusion, the content recommendations aim to ameliorate the experience of watching TV.

In this context, four stakeholder groups can be identified:

- On the producer side: video conductor[6] and director.
- On the consumer side: two types of viewers, viz active and passive users. Users are classified as active, if their TV watching mode can be described as lean forward otherwise they are considered as passive users.

As starting assumptions, the final recommender system will be realised within the LIVE project. A prototype for testing the algorithm and the prediction quality is built. Video material and metadata descriptions are not yet available from the other project parts. An adequate substitute for this data has to be found. The crucial part will be integrating live measurement of audience data.

5.1.2 Problem Scenarios

Problem scenarios address the basic requirements of this project. The scenarios have been derived from current TV usage behaviours.

5.1.2.1 Holly Cooper

Holly is a mother of two children (10 and 13 years). She is 41 years old. She works half-time as a biology teacher. She loves to work in the garden and to walk around. As she has very little free time, she only watches TV while ironing or working out at home. The remote control is always near by. If the TV programme was not attractive to her, she would zap around. For her, TV is a background distraction while doing something else. Usually, it does not play any role elsewhere. Sometimes, she watches a DVD with her husband Dean.

[6] A video conductor can be described as a super-director who decides among different streams provided by other directors. For more information cf. Wages et al., 2007.

5.1.2.2 Seth and Summer Middleton

Seth and Summer are a newly wed couple. They are both end twens. They are planning to have children in the future but that is not yet acute. Both of them have finished their studies some time ago and have good jobs now. They could be classified as DINKYs[7].

They like to watch football together, but except football Summer is not much into sports. She prefers to watch movies and TV series, e.g. she never misses a single episode of The O.C. Contrarily, Seth loves to watch sports. He loves to watch rugby and some other sports, but he hates to watch athletics. Furthermore, he loves to read comics.

Usually, they have a long discussion when starting to watch TV. When they have managed to agree on a programme they keep watching it.

5.1.2.3 Ryan Ashbrook, Tate McKenzie and Jeff Cohen

Ryan, Tate and Jeff have been best mates for a long time. They like to go out for a beer. Every Saturday afternoon, they meet in their favourite pub to watch the weekend's football games. They enjoy watching sports with many other people. They prefer public viewing events over watching sports on their own. However, if meeting up does not work out, they are still willing to watch sport events solitarily.

5.1.2.4 Frank Doose

Frank works in an established assurance as a salesman. He mainly watches TV when coming home from work. He is not really interested in the programme, so that he switches forth and back. But if he switched to a channel he likes he would stop zapping. He combines watching TV with having dinner. Having finished dinner, he usually turns TV off.

[7] Double income no kids yet

5.1.2.5 Luke Roberts and Niki Bilson

Luke and Niki have been a couple for 1.5 years. They really enjoy being together. They only watch TV events that they have scheduled beforehand. They usually watch the shows from the beginning until the end. In their leisure time, they love to do sports, go for a walk or dine in a restaurant. The most important thing for them is being together.

5.1.3 Problem Claims

The problem scenarios described in chapter 5.1.2 are transformed into problem claims. These claims are the essence of the scenarios.

Situation Feature	**Possible Pros (+) or Cons (-) of the Feature**
Viewer chooses programme via remote control	+ Viewer controls the programme he watches + Programme can change over the time → programme does not become boring - Viewer has to stop his other work (if TV is used as background distraction) - Remote control has to be near by
Different programmes defined by creators	+ Everyone might find something + Different user habits, e.g. watching one channel, zapping etc., supported + No approval of a channel by the viewer necessary - If the programme changes the user has to react - Flow of the channel externally predefined
Using TV as background distraction	+ Viewer gets information or entertainment while doing something else - Content most likely does not suit user needs

Fig. 2: Problem Claims derived from Problem Scenarios

5.2 Design

5.2.1 Activity Scenarios

5.2.1.1 Holly Cooper

Holly, a biology teacher, has just come home from school. It was a rather hard day as her 10 -12 year old students were quite upset. She plans to do some work in the garden as recreation later in the afternoon, but first she has to do some housework. She turns the TV on and switches to her personal recommendation channel. The System recommends her to watch a video about neurobiology. She loves to watch videos that have to do with her job so she turns it on.

In the evening her husband Dean comes home. They decide to watch a movie together. Dean bought a new DVD yesterday that they decide to watch.

5.2.1.2 Seth and Summer Middleton

Seth and Summer have been out for a walk. While preparing dinner they argue what to watch on TV. The personalised recommender proposes the half final of the 2007 Rugby World Cup between England and France to Seth and the new episode of The O.C. to Summer. As the rugby half final is not attractive to be watched afterwards they decide to record the The O.C. episode and watch rugby.

5.2.1.3 Ryan Ashbrook, Tate McKenzie and Jeff Cohen

Ryan, Tate and Jeff meet at their favourite pub to watch football together as they do every Saturday afternoon. As they rush in, they identify themselves to the viewing system and form a group of viewers. The system builds a personalised sports stream in accordance to the group profile.

5.2.1.4 Frank Doose

Frank has just come home from work. He turns on his TV and switches to his personal channel and prepares his dinner. First the system shows him the most recent news show. After the news, the system recommends him an episode of a comic series. He accepts the recommendation and watches the series.

5.2.1.5 Luke Roberts and Niki Bilson

Luke and Niki are planning their weekend. They start the Electronic Programme Guide (EPG) of their television receiver to see what films will be broadcast this evening. One of their favourite movies is scheduled for tonight. They plan to watch that movie and meet some friends in a pub afterwards.

5.2.2 Activity Claims

The activity claims described in Fig. 3 have been derived from the activity scenarios of chapter 5.2.1. They serve as a basis for the UML Use Cases described below.

Situation Feature	**Possible Pros (+) or Cons (-) of the Feature**
Smart Zapping	+ Reduces required user interaction + When watched programme is finished, new content is available + Personalisation + Real lean-back - Could be distracting - No control about what comes next - Recommendations can be intransparent or even inaccurate - only suitable for one viewer
Tag Channel	+ Viewers can choose a distinct area of interest - Only possible with IPTV (at this moment)
Grouping	+ Personalisation for a group of users - Average channel for a group could lead to bad recommendations for the specific viewer
Video on Demand	+ Time independent access to video - Only possible with IPTV (at this moment)
Default Programme	+ Fallback solution

for unknown viewer	+ Average channel + Preserving the status quo + No need to register

Fig. 3: Activity Claims derived from Activity Scenarios

Four activity claims have been derived from the respective scenarios. They serve as a basis for further development. However, only one claim is taken into account: smart zapping and the necessary recommendations. The three other features form interesting viewpoints but lead away from the primary focus.

Smart zapping means that the system changes automatically to another asset if it suits better with the viewer or the current asset is finished. This feature is derived from the last.fm recommender system described in chapter 2.7.3.

5.2.3 Use Cases derived from Scenarios

In contrast to Scenario-Based Development as introduced by Rosson & Carrol the scenarios do not serve as a basis for development of the real application. They are used as a means of requirements specification for the algorithm. Consequently, the scenarios are used to clarify the problem scope and are then being transformed into UML Use Cases.

The UML diagrams contain the essence of the adjacent scenarios. If an interaction scheme already existed in another Use Case it would be left out. Consequently, no specific Use Case is derived from the Frank scenario. Fig. 4 to Fig. 7 illustrate these Use Cases.

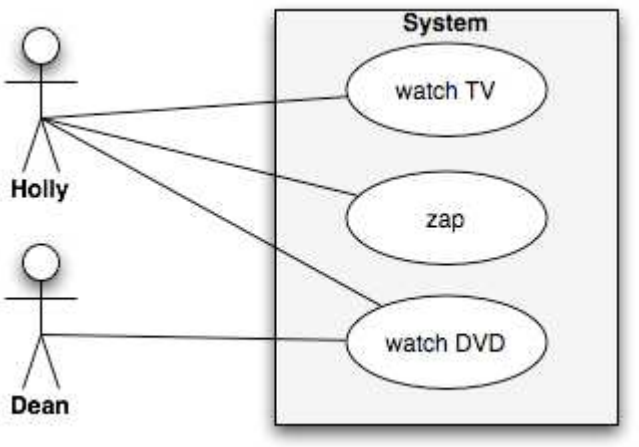

Fig. 4: UML Use Case derived from Holly scenarios

Holly watches television mainly as distraction and zaps through the channels. Together with Dean, she also watches DVD (Fig. 4)

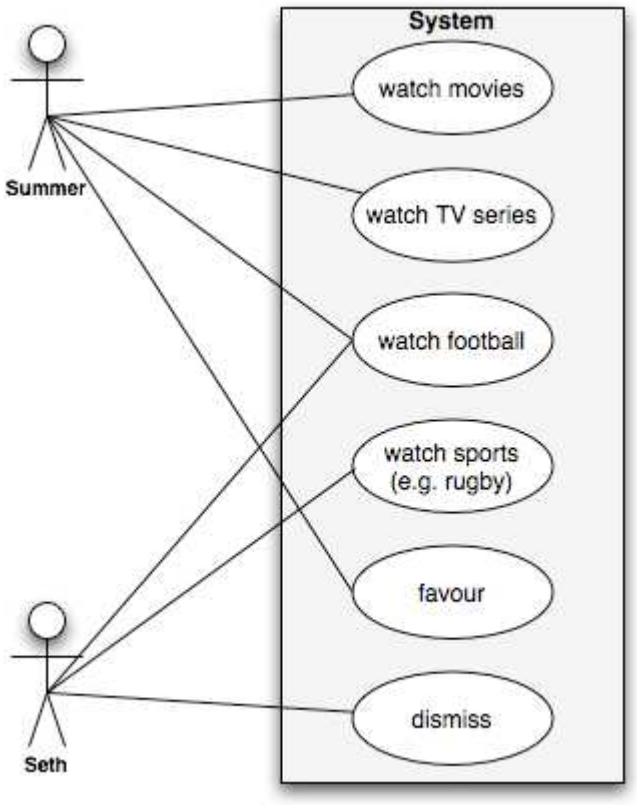

Fig. 5: UML Use Case derived from Seth and Summer scenarios

Summer and Seth (Fig. 5) like to watch several TV formats together. Some of the formats they like both, some one of them does not like them at all (dismiss) and some one of them really likes (favour) them.

The Use Case derived from the Ryan, Tate and Jeff scenarios (Fig. 6) only mentions the new actions identify with the system and form a group.

As the Frank scenarios do not provide any new information, no Use Case is created based on these scenarios.

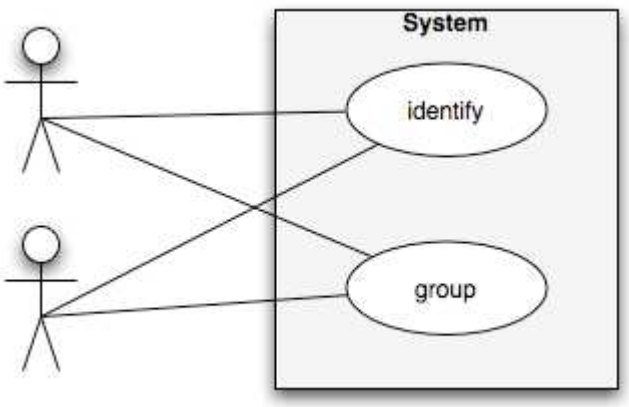

Fig. 6: UML Use Case derived from Ryan, Tate and Jeff scenarios

In contrast, a new feature can be derived from the Luke and Niki scenarios. They plan their TV programme in advance, i.e. they should be supported by a scheduling mechanism (Fig. 7).

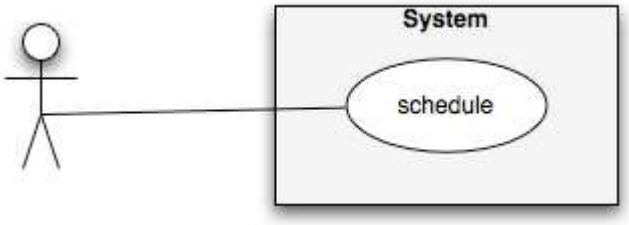

Fig. 7: UML Use Case derived from Luke and Niki scenarios

5.2.4 Resulting Use Cases

All Use Cases developed in chapter 5.2.3 have been combined and simplified to one single Use Case shown in Fig. 8. It illustrates the features that can be derived from the scenarios developed in this chapter.

This work is mainly concerned with the proposal of an algorithm that calculates predictions and recommendations for media assets, so that all actions concerned with user management, i.e. identify and group, are left out. Furthermore, the Use Cases do not map any relationships between the users. Consequently, the algorithm is developed to calculate recommendations for only one user at a time. Furthermore, scheduling is also not developed. However, scheduling a programme could be mapped with favouring it. All programme items are mapped to one abstract programme item.

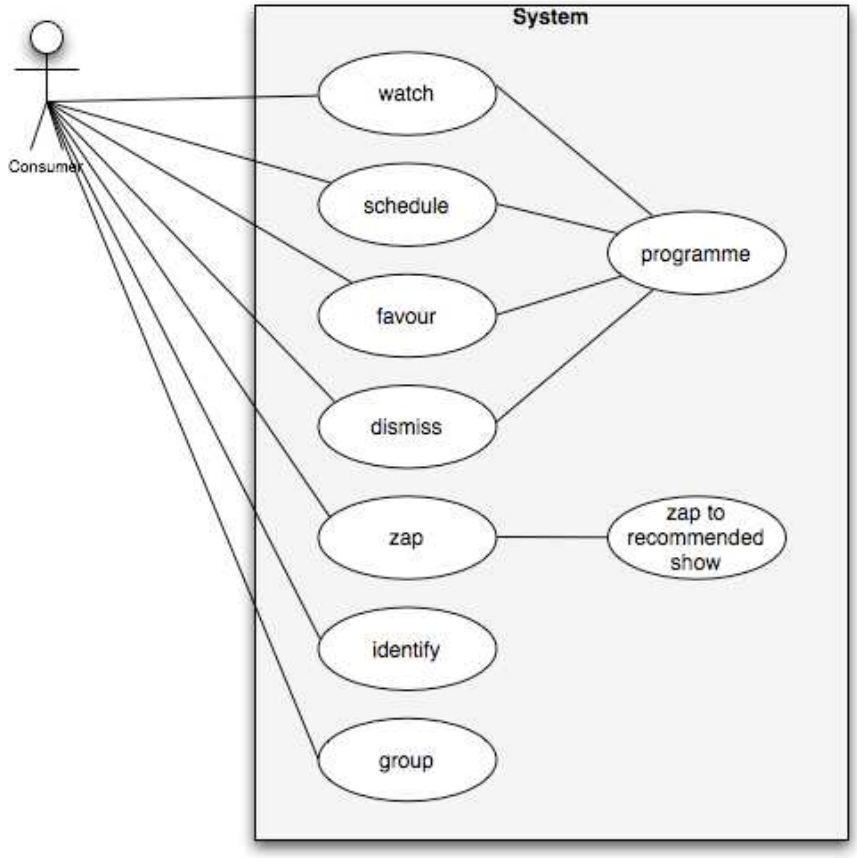

Fig. 8: Combined Consumer UML Use Case

Fig. 9 describes the tasks the system is to perform. Those tasks have not been explicitly formulated within the scenarios but have been inferred from the inherent implications.

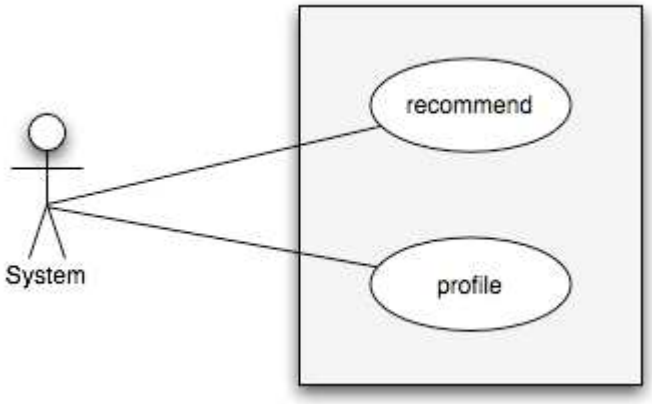

Fig. 9: Use Case illustrating the actions the system is to perform

5.3 Prototyping and Evaluation

The process of prototyping and evaluation is mainly addressed in chapter 10. Chapters 6 to 9 describe the development of a recommendation algorithm based on the requirements analysis in this chapter. However, an evaluation of the whole system with more users is scheduled in mid 2008 within the LIVE project.

6 Realisation of Viewer Profiles using Implicit and Explicit Feedback

A crucial aspect for calculating recommendations is accurately describing the user and his relations to the media items $i \in \mathfrak{I}$ [8]. These user profiles are built using two means of user feedback derived from the scenarios and Use Cases developed in chapter 5: watching statistics (i.e. watching an asset or zapping) and asset rating on a dual scale (i.e. favour and dismiss).

6.1 Mapping Feedback to a Benchmark Variable

Watching statistics form a variation of implicit feedback whereas asset rating is a means of explicit feedback.

Watching an asset can be mapped on three characteristics: watched, zapped and not watched at all. Watching an asset is interpreted as positive feedback whereas zapping has negative connotations. Not watching an asset is treated as neutral feedback.

[8] $\mathfrak{I}$ is the set of all items available to the viewer. Media items are also called media assets in this thesis.

Rating an item can be expressed in two ways: favour the item and dismiss. Rating explicitly implies that a user has watched the item. Dismissing the item induces zapping to the highest recommendation.

Both means of feedback are mapped to a benchmark variable as indicated in Fig. 10. Mapping two different types of feedback to one single variable can be seen as losing information. As the higher absolute values also include the weaker feedback this is not the case. In contrast, using only one variable simplifies the whole process. Furthermore, this mapping is bijective so that another weight for the feedback can be simply implemented. In this examination, both means of feedback are treated as equivalent, but this can easily be changed.

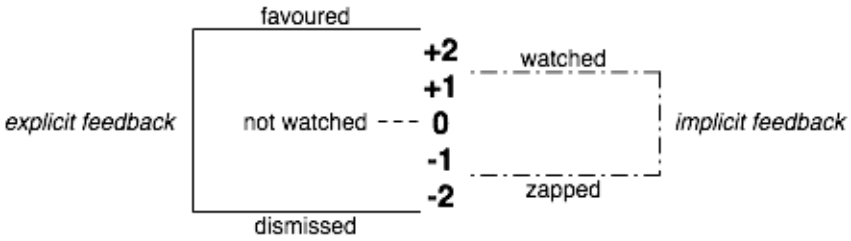

Fig. 10: Values of the benchmark variables used to describe watching and rating of live media assets

The feedback is always mapped to the whole asset, no matter whether the respective action is performed at the beginning of the asset or at its end.

6.2 Possible Enhancements of Feedback Collection

Further enhancement to the quality of implicit feedback could be derived from mood analysis methods using real time face analysis as introduced by Küblbeck (Küblbeck, 2007). Mood analysis could provide more relevant information about how a user likes an asset without requiring him to perform an action. Even though no evaluation of the face analysis method is available at the time of writing, it seems to be an auspicious approach.

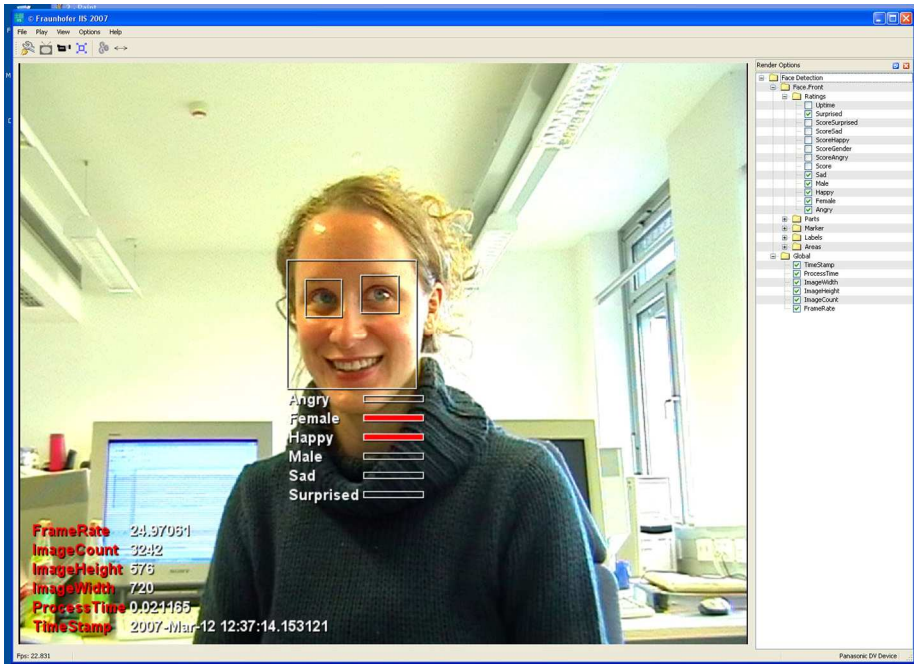

Fig. 11: Screenshot Face Analysis Software (Küblbeck, 2007)

How such enhancement could be integrated in a feedback mechanism depends on the characteristics it provides. One possibility derived from Küblbeck's proposition would be to integrate the indicators for angriness, happiness and sadness. As solely a demo version of the software is available this cannot be tested yet. Hence, it is not integrated in the proposed algorithm.

6.3 Weighting Media Assets according to Feedback

Every user takes a specific path through the channels provided to him. He rates every item he watched as described in chapter 6.1. One example is illustrated in Fig. 12.

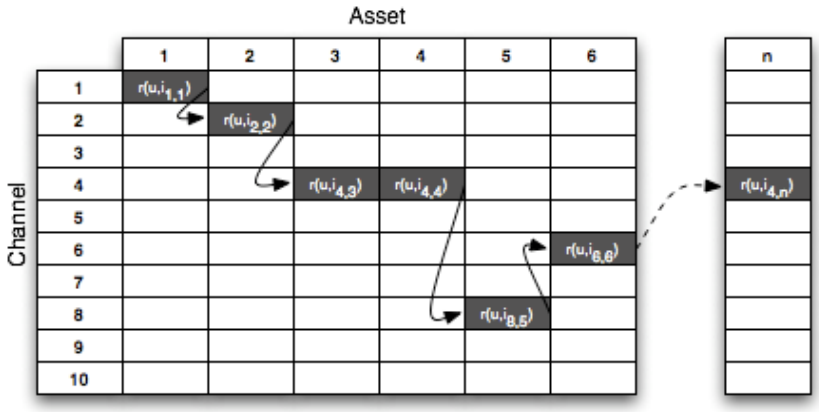

Fig. 12: Example and related Ratings for User u watching an interactive Video: Black indicated items have been watched.

Each media asset has to be weighted according to the feedback the consumer provided. A simple tagging mechanism is used to describe the media assets. One asset can (theoretically) have an unlimited number of tags, but must have at least one. The set containing all tags is denoted with $\mathfrak{T}$.

The rating of an asset is mapped on the asset's tags, i.e. the asset tags are rated with the same value as the whole asset. If a tag is not associated with an asset, it will be implicitly rated with 0.

Tags are used as a means of describing media assets because they are a rather easy to implement form of metadata. As described in chapter 3.3 it is impossible to predict the contents of the next assets. It is also quite difficult to calculate profile similarity only according to the seen assets as they can only be seen when they are broadcast.

6.4 Building User Profiles

User profiles describe the users by the tags associated to all assets. In a first step the tag vector $T_{i,t}$ (9) is created. The tag vector describes whether tag *t* is associated with item *i*.

$$T_{i,t} = \begin{cases} 1 & \text{if } t \in \mathfrak{T} \text{ in tags associated with asset } i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In the next step, the set $I_{u,t}$ of all items tagged with *t* and rated by user *u* is defined (10).

$$I_{u,t} = \left\{ i \in \mathfrak{I} \middle| r_{u,i} \neq 0 \wedge T_{i,t} = 1 \right\} \tag{10}$$

The user profile $U_{u,t}$ for user *u* and tag *t* is calculated as arithmetic mean of the ratings user *u* gave to items associated with tag *t* (11). If *u* did not rate any item tagged with *t*, $U_{u,t}$ would have the value of 0.

$$U_{u,t} = \begin{cases} \dfrac{\sum_{i \in I_{u,t}}^{i} r_{u,i}}{\# I_{u,t}} & \text{if } \# I_{u,t} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

7 Content-Based Predictions

Content-Based predictions are one possible means to calculate recommendations for a user. These recommendations are built on the user profile and the tagging mechanism described in chapter 6.

7.1 Prediction Algorithm

A Content-Based prediction for one user and one asset is realised in a quite simple and straightforward manner:

1. A filtered user rating vector is created from the user profile and the tags associated with the current asset: the mean ratings of the active user (given in the user profile) associated with the tags of the actual asset form that vector.
2. The arithmetic mean of the filtered vector is calculated. It is treated as a Content-Based prediction.

A python example implementation is given in Fig. 13. The vector `userprofile` is the profile calculated for the current user as described in chapter 6.4. The vector `assettags` contains all tags associated with the asset.

```
def cb_prediction(userprofile, assettags):
    # the vector userprofile contains the profile values of
one user associated with a tag
    # the vector assettags contains all tags associated with
an asset
```

```
    return mean([userprofile[tag] for tag in assettags])
```

Fig. 13: Content-Based Prediction as implemented in the test application

Two other methods have been discussed for creating Content-Based predictions: using a Pearson correlation and using a dot product of the user ratings vector and the tag vector. For both cases the tag vector is defined in equation (9).

The first approach to calculate a Content-Based prediction uses a Pearson correlation to measure the linear distance between the profile and the asset. This approach would have yielded to more steps to be performed during calculation. However, a more important trade-off would have been that the tag vector must be filled with many zeros to describe tags that are not associated with the asset. This induces that the correlation value is very low even though the tags might be the best liked in the profile. On the other hand you could consider only taking tags into account that are associated with that item.

Calculating a dot product of the rating and tag vector, as used in Lifetrak (Reddy & Mascia, 2006), yields to the same result as summing up the values of the tags associated with the asset. This result must be mapped to the rating interval by either dividing by the number of tags total or the number of tags associated with the asset. Dividing the result by the number of tags associated with the asset is equivalent with the algorithm proposed above. Equation (12) shows the prediction calculated as dot product (with *n*: number of tags associated with the asset).

$$c_{a,i} = \frac{T_i \cdot U_u}{\sum_{t \in \mathfrak{T}} T_{i,t}} = \frac{\sum_{t \in \mathfrak{T}} T_{i,t} U_{u,t}}{\sum_{t \in \mathfrak{T}} T_{i,t}} \tag{12}$$

Fig. 14 illustrates the process how a content based prediction is calculated from consumer actions.

Fig. 14: Process Diagram of the Content Based Predictor

7.2 Reliability of the Content-Based Predictor

According to Melville et al. (Melville, Mooney & Nagarajan, 2001), the factor $l_{\min}$ – the minimal number of training items to deliver good predictions – is calculated by using a learning curve for the Content-Based predictor. On this learning curve, the mean absolute error[9] *MAE* is plotted against the number of training items *n*. A minimal *n* (l_{min}) has to be found to describe the predictor as trusted under the condition that the curve is monotonic decreasing. The Content-Based predictor is trusted when the derivative value of the curve is greater than -0.001 (13). Hence, they use a high significance level.

$$MAE'(l_{\min}) \geq -0.001 \tag{13}$$

In this case, as discrete numeric values are examined, the derivative is defined by the difference quotient (14).

$$MAE'(n) = \frac{MAE(n+1) - MAE(n)}{1} = MAE(n+1) - MAE(n) \tag{14}$$

Melville et al. calculated the learning curve for 132 users with at least 200 items rated. It is not yet possible to say how many users are necessary to test this algorithm as the final probe design being carried out in the 2008 field trial is not yet described.

[9] The mean absolute error is defined further on in chapter 10.4.1.

7.3 Pseudo User Ratings Vector

Content-Based predictions are calculated for every asset available at a certain time. These calculations are performed for the active viewer, i.e. the target viewer of the prediction, and all offline viewers. They serve as a basis for collaborative filtering by forming a pseudo user rating vector for the assets available at that time as introduced by Melville et al. (Melville, Mooney & Nagarajan, 2001) and described in chapter 2.5.

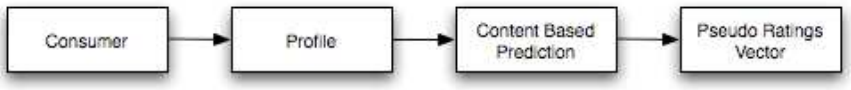

Fig. 15: Process Diagram of the Content Based Predictor including the Pseudo Ratings Vector

Fig. 15 shows how the pseudo ratings vector is integrated in prediction process.

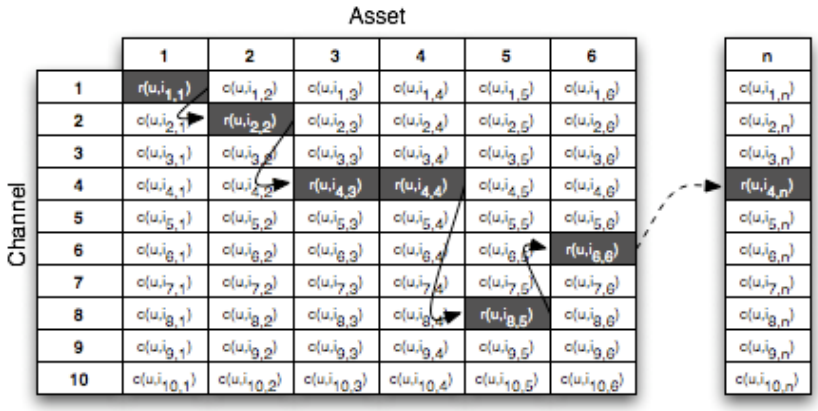

Fig. 16: Pseudo User Ratings for one User (cf. Fig. 13)

Fig. 16 illustrates the pseudo user ratings for one example viewer. The Content-Based predictions are calculated according to the profile that has been built up at this time.

8 Collaborative Filtering for a Hybrid Recommender

As hybrid recommendation algorithms usually perform better than either collaborative filtering or Content-Based filtering alone (Melville, Mooney & Nagarajan, 2001; Melville, Mooney & Nagarajan, 2002; Adomavicius & Tuzhilin, 2005), a hybrid approach on recommendation is also tested in this case. However, which approach performs better has to be evaluated for this case, too.

8.1 Watching Time Weighting

The relevance of the all users for the predictions is calculated according to the number of assets a user watched in the recent time. A viewer that is watching the channel should be weighted higher than an offline consumer. It seems that after 2 minutes the user has decided to either watch or to stop watching the programme. This assumption is also supported by the feedback the users gave after the tests described in chapter 10. However, this cannot be proven yet as the data does not suffice, so that the properties of the factor are described in this chapter.

The time weighting factor tw_u (15) is calculated like the weighting factors introduced by Melville et al. in chapter 2.5. In this equation n_u refers to the number of items user u has watched in the recent time, i.e. since starting to watch in this session.

$$tw_u = \begin{cases} \frac{n_u}{l_{\min}} & \text{if } n_u < l_{\min} \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

As the necessary data for generating this learning curve, as described in chapter 7.2, is not yet available, a value of 1 is used for $l_{\min}$, i.e. the value of tw_u is always 1 except if the user did not watch any item. This means that no time weighting will be available in this examination.

Whether this factor leads to better recommendations should be evaluated when the necessary data is available.

8.2 Calculating Predictions

The prediction algorithm used in this work is inspired by the prediction algorithm used by Melville et al. The confidence factors used in Content-Boosted Collaborative Filtering (cf. chapter 2.5) are replaced by the time weighting factor described in chapter 8.1.

$$p_{a,i} = \bar{v}_a + \frac{tw_a(c_a - \bar{v}_a) + \sum_{\substack{u=1 \\ u \neq a}}^{n} tw_u P_{a,u}(v_{u,i} - \bar{v}_u)}{tw_a + \sum_{\substack{u=1 \\ u \neq a}}^{n} tw_u P_{a,u}} \tag{16}$$

The predictions $p_{a,i}$ for the active user a and item i are then sorted descending. The prediction value for the item the user is watching is replaced by the respective rating. The asset with the highest predicted rating is then recommended to a.

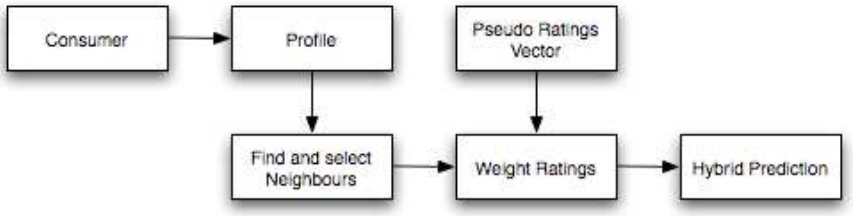

Fig. 17: Process Diagram of the Hybrid Predictor

Fig. 17 illustrates the process of the hybrid predictor that is proposed in this chapter.

9 Analysing and Interpreting Predictions

In the live broadcasting context, as already mentioned in chapter 3.3, it is impossible to predict the contents of the assets that are to be broadcast and the behaviour of the neighbours in the future. Consequently, a replacement for this data has to be found.

For this work, it is assumed that the contents of two sequential assets in one channel do not differ significantly. As a consequence, the next asset is described with the description of its preceding asset.

Recommendations will only be useful if the prediction is better than the rating of the actual asset. Due to the live context of the recommender system, a viewer can only watch one of the assets and not both. Consequently, an item will only be recommended in this case.

A better prediction can occur in three different cases:

1. The viewer is watching an item and did not favour it (actual item rating 1)
2. The viewer zapped to another channel (actual item rating -1)
3. The viewer dismissed the actual channel (actual item rating -2)

Case 2 is treated as an intentional decision for another channel, so that no supplementary recommendation would be necessary. Consequently, the system should only recommend another asset in cases 1 and 3.

10 Evaluation of the proposed Algorithm

The algorithm described in chapters 6 to 9 is evaluated with the help of a small test application and data derived from watching the interactive DVD "Vision Europe" produced in the EU IST Project MECiTV[10]. This interactive video has been tagged with keywords describing the media assets. The recommendations are compared with the actual values of the benchmark variable.

10.1 MECiTV: Vision Europe

Vision Europe provides ten different parallel channels. As shown in Fig. 18 several editorial switching points are defined. At each switching point, another channel is recommended to the user.

In the test case, a media asset is defined as a 10 second part of the live stream, i.e. the first asset ranges from 0-9 sec, the second from 10-1 9s etc. These parts do not necessarily have the same content, i.e. scenes are not taken into account. For practical reasons they are treated as if having one major subject. In a future implementation, these media assets should be defined more accurately according to scenes and realistic changes. To describe the assets more accurately, cut detecting algorithms

[10] MECiTV was a research project that aimed to combine media content with media technologies. It was one of the predecessors of the LIVE project. More information about MECiTV can be obtained on http://www.meci.tv

could be used. However they should not be longer than a defined amount of time in order to maintain comparability of those items.

Notwithstanding, the test case is rather far away from the real usage scenario. One problem of the case is that not much test users have performed the test but this could also be an implication in the real usage scenario. The bigger problem is indeed, that no real recommendations (as discussed here) but editorial switching points are shown to the user. As MECiTV is not very exciting, the test viewers will not enjoy the test. Hence, the data will not be very resilient. However, testing the algorithm is necessary and MECiTV is the only available substitute for live data at this time.

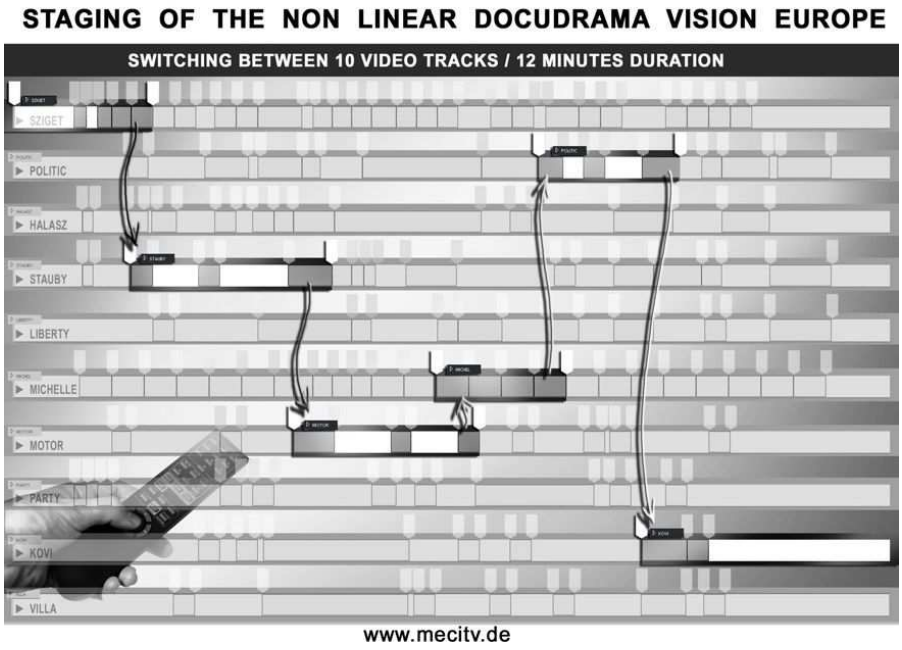

Fig. 18: Channel overview of the MECiTV DVD "Vision Europe" (MECiTV Consortium, 2004)

However, the tests are conducted with a very small amount of users. As the recommendation algorithm is also intended to work with very little data, this can be seen as a suitable trade-off. Using substitute data and testing with a small number of users

induces the necessity of retesting the algorithm with real user data acquired later within the LIVE project.

10.2 Test Application

The test application is built in python using the object serialisation module pickle and the GUI toolkit Qt3 (with its interface PyQt3). Pickle provides object storing to and restoring from files with one single function call. For that reason, no overhead for defining a database or XML scheme is needed. Qt3 and its python interface PyQt are available on Linux, Mac OS X and also on Windows. The Qt library is used to develop the Linux desktop KDE, the Opera Web browser, Adobe Photoshop Album, the last.fm player and several other software products.

Programmes and especially mathematical algorithms are rather easy to implement in python as its programming paradigm supports object orientation as well as functional programming.

The scope of the test application is: annotate the media assets (as defined above), record usage statistics, compute recommendations, log them and calculate evaluation meters defined in chapter 10.3.

The implementation of the algorithm is checked using unit tests. Unit tests are a flexible but powerful means to test algorithms. They have proven to be very successful within software development.

Fig. 19 shows how the test application is realised. The four components consumer, asset repository, asset management and channel are grouped into the profiler module. The module collects information about the assets and the consumer, i.e. it logs which assets the user watches and how he votes on them. This data is persistently stored on the hard disk. In the test environment the test conductor uses the profiler application (cf. Fig. 20) to annotate the media streams and as a logging means. The MECiTV DVD and the test environment cannot communicate directly, but they use timestamp and channel to identify an asset.

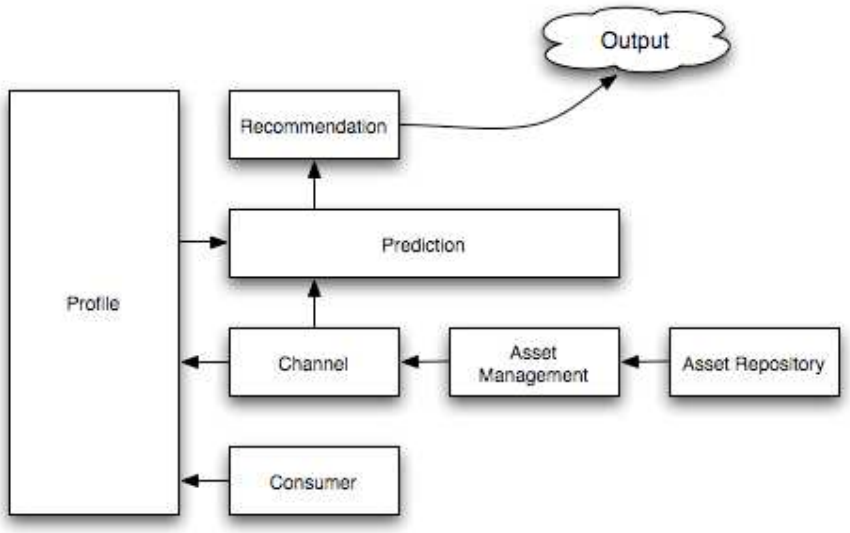

Fig. 19: Layer model of the test environment architecture

The profile module calculates the user profiles from the user data. The prediction module calculates predictions of user ratings based on the user profiles and the channel profiles.

From these predictions the recommendations module builds weighted lists of channels for every 10 second item. These recommendations are evaluated with the metrics described in chapter 10.3. The recommendation module creates CSV files containing the different metric values.

A data maintenance tool standing outside of this hierarchy enables data related actions such as exporting data to XML[11] or CSV files, deleting tags or rebuilding the database. A more detailed description of this tool is also included in the appendix.

[11] The XML schemas are included in the appendix

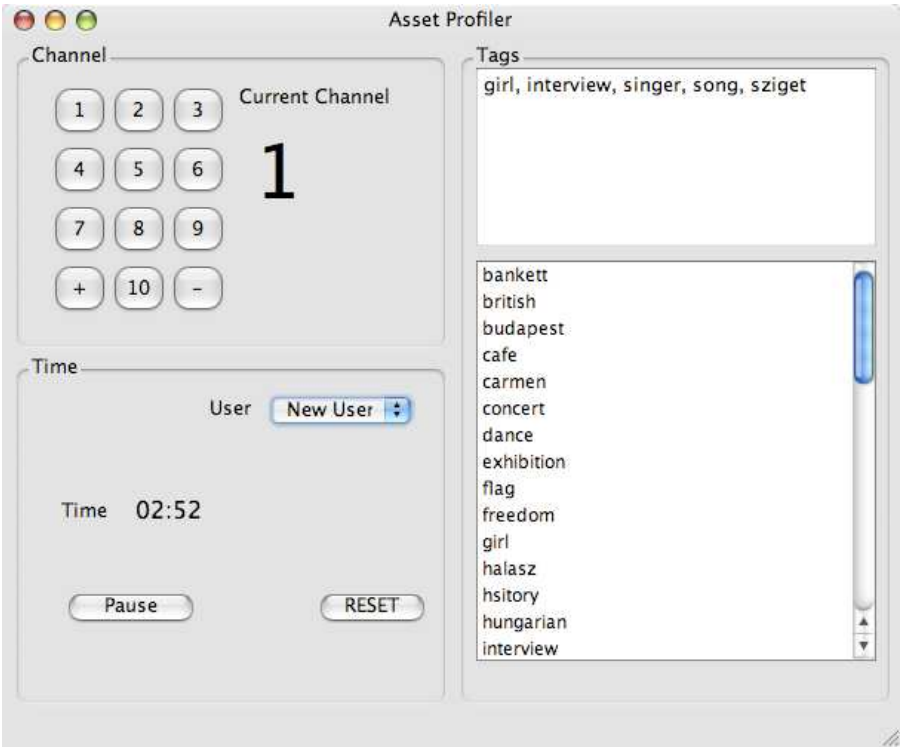

Fig. 20: Screenshot of the Profiler Application

The test application and the data gathered during the user experiments are enclosed on the CD bundled to this thesis. Information about how to install and use the application can be found in the appendix.

10.3 Tagging the Media Assets

The tagging process has been conducted in a straightforward manner. While watching the channels from beginning to the end, keywords have been assigned to the media assets using the profiler application. If a similar keyword has already been assigned to another asset, this keyword was used.

The switching points described in chapter 10.1 could be indicated with special tags, viz recTarget and recSource. A tag could also indicate the two types of recommenda-

tion, viz recTimed and recStressless. As the types of recommendations are not evaluated within this thesis this has not been done. An evaluation of the types of recommendations would only make sense if a big number of test persons would be observed.

10.4 Metrics for Prediction Accuracy Evaluation

The algorithm is evaluated through two different metrics that are commonly used within recommender system research: the statistical accuracy metric mean absolute error (MAE) and the decision support measure Receiver Operating Characteristics (ROC). Both metrics compare the actual rating a user gave to an item with the predictions made by an algorithm. (Melville, Mooney & Nagarajan, 2001; Melville, Mooney & Nagarajan, 2002)

10.4.1 Mean Absolute Error

Statistical accuracy metrics evaluate predictions by comparing them with actual ratings. The mean absolute error is defined as the mean absolute difference between prediction $p_{u,i}$ provided by the algorithm and the actual rating $r_{u,i}$ user u gave to item i (17).

$$MAE = \frac{\sum_{u=1}^{n}\sum_{i=1}^{m}\left|p_{u,i} - r_{u,i}\right|}{n \cdot m} \tag{17}$$

The lower the mean absolute error, the better an actual predictor can be classified.

10.4.2 Receiver Operating Characteristics

Receiver Operating Characteristics, as decision support measure, are used to describe how predictions support the users in selecting high-quality items. (Melville, Mooney & Nagarajan, 2001) A high rating is usually mapped on accepting an item whereas a low rating is mapped on rejecting it. As the benchmark variable defined in chapter 6.1 already describes accepting, i.e. watching an item, and rejecting, it can

directly be used. Accepted items are items that are rated with minimum 1. All other items are rejected.

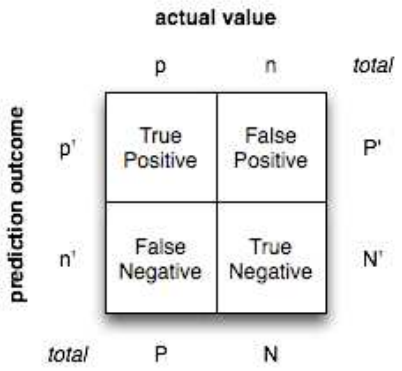

Fig. 21: 2×2 contingency table as used for Receiver Operating Characteristics

Fig. 21 illustrates the four categories a prediction using a binary scale: The outcomes and the actual values can either be positive (p), i.e. accepted, or negative (n), i.e. rejected. Consequently, there are four possible outcomes overall. If the outcome of a prediction and the actual value are both p this will be called a true positive (TP). However, if the prediction's value is p and the actual value is n this will be called a false positive (FP). Vice versa, a true negative (TN) will occur if the prediction outcome and the actual value are both n. A false negative (FN) will occur if the prediction outcome is n and the actual value is different, namely p. The sum of true positives and true negatives is the number of accurate predictions. (Wikipedia, 2007d; Fawcett, 2004)

In this analysis, three measures are observed: true positive rate (TPR), i.e. hit, false positive rate (FPR), i.e. false alarm, and accuracy (ACC). The measures are defined in equations (18)-(20).

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{18}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{19}$$

$$ACC = \frac{TP + TN}{P + N} \tag{20}$$

The best possible prediction would yield in a TPR of 1 and an FPR of 0 and consequently in an accuracy of 1. A TPR of 0.5 is equivalent to random guess, so that a robust algorithm must be better than random guess. A worse algorithm is consequently worthless.

10.4.3 Analysis Ranges

Both metrics are calculated for distinct moments on the time scale to illustrate how predictions improve within the experiment. They are calculated for the Content-Based predictor alone and the hybrid predictor.

Two analysis ranges should be used for evaluation: the first range comprises all assets the test user has seen. Consequently, predictions will be calculated for every asset no matter if they have been recommended or not. The second range only takes assets into account where recommendations are available. Both ranges are illustrated on a time scale.

10.5 Test Setup

The main goal of the tests is to gather usage data of an interactive TV platform. Hence, the setting is not thoroughly designed to produce comparable data. If any usability problems occur, they are noted but the tests are not intended to be a usability evaluation.

10.5.1 Test Users

The tests have been conducted in October 2007 with 13 employees of Pixelpark Agentur Köln. 10 test persons are masculine, 3 feminine. 9 persons are 20-29 years

old, 2 are aged between 30 and 39 years and 2 older than 40 years. A detailed overview about the characteristics of the test persons is given in Fig. 22.

	20-29 yrs	**30-39 yrs**	**40+ yrs**	***total***
Masculine	8	0	2	*10*
Feminine	1	2	0	*3*
total	*9*	*2*	*2*	*13*

Fig. 22: Overview about Test Persons of the Recommender Application

10.5.2 Test Environment

Fig. 23: Setting as used for the User Tests

The tests have been conducted in a quite straightforward manner. The test person watched the interactive video on one computer while a test conductor logged the actions of the test person with the aforementioned asset profiler on another computer.

The test conductor first explained the test person how to use the MECiTV application, i.e. how recommendations work, how the remote control is supposed to be used

and what other means of interaction are available. The difference between real personalised recommendations and the editorial switching points used in MECiTV is also explained to the user. Anymore, the test viewer is told to give feedback about dismissal and favour of a channel vocally. This vocal feedback is logged by the test conductor.

In the next step the test users watch and interact with the video as intended by the authors. During the test session they can also give feedback about how they feel while watching the movie. However, this feedback is not formally recorded but noted as a resume.

10.6 Results of the Evaluation

10.6.1 User Reactions

The reactions of the test persons about MECiTV are resumed in the following part. This resume is no formal usability evaluation but a collection of feedback given by the test viewers. It is not representative and the feedback has not been journalised during the test.

The most important problem occurring during the tests was, that the contents of the interactive docudrama did not attract any of the viewers. They were very dissatisfied about its poor dramaturgy and tension. Most of them described the video as boring. Some of the test viewers said that they would have switched off after a few minutes if they had not been in a test situation.

The watching experience suffers from a more or less bad implementation of the storyboards: some switching points do not yield to the channel that are announced in the video but to a seemingly random other channel. This evoked disorientation and annoyance among the viewers. As a consequence, they stayed at a rather random or in the best case least uninteresting channel or just randomly zapped through the channels.

The mode of recommendations was also not understood by all users even though it was explained during the preparation of the test session and within the video. As clicking the ok button always leads to another channel, some test persons thought that a recommendation is always available. In fact this behaviour was simply due to the erroneous implementation of the interactive video.

Some users tended to zap rapidly, whereas other viewer tended to stay within one channel they had previously chosen. After a certain amount of time most users reduced the number of zaps and tended to stabilise on one channel. Favouring and dismissing the channels worked but was sparsely used. The test users only used this feedback mode when really loved or really hated the channel. It can be the case that this feedback mode has not been used as it was not implemented within the user interface but required external feedback.

Consequently, the threshold for ROC should have two values: as proposed afore a value of 1 and additionally a value of 0.5. Hence, another item would also be recommended if its prediction would be better then 0.5.

Furthermore, another value for favouring and dismissing, e.g. 5 and -5, an item could be used. Whether this approach would perform better is not tested in this thesis, as the low usage of favour and dismissal could also be caused by the suboptimal interface. However, this can be easily changed within the predictor application by adjusting the parameters as described in the appendix.

Consequently, the data gathered within these tests is not very resilient and the algorithm has to be retested when the necessary data is available in mid 2008 from the Beijing 2008 Olympic Games field trial. However, this trial can give hints about the quality of the recommendation outcomes in spite of the suboptimal test design.

As consequence of the feedback, a high focus should be laid upon a high content quality as well as on a good dramaturgy during the LIVE test scenario in mid 2008. This should reduce arbitrary reactions of the test users.

10.6.2 Analysis Ranges

The analysis ranges introduced in chapter 10.4.3 have been changed as all users went different paths through the interactive video (cf. Fig. 24). Consequently, switching points were available at different times, so that comparing predictions for assets where switching points are available does not work. Hence, predictions are only calculated for every position in the timeline.

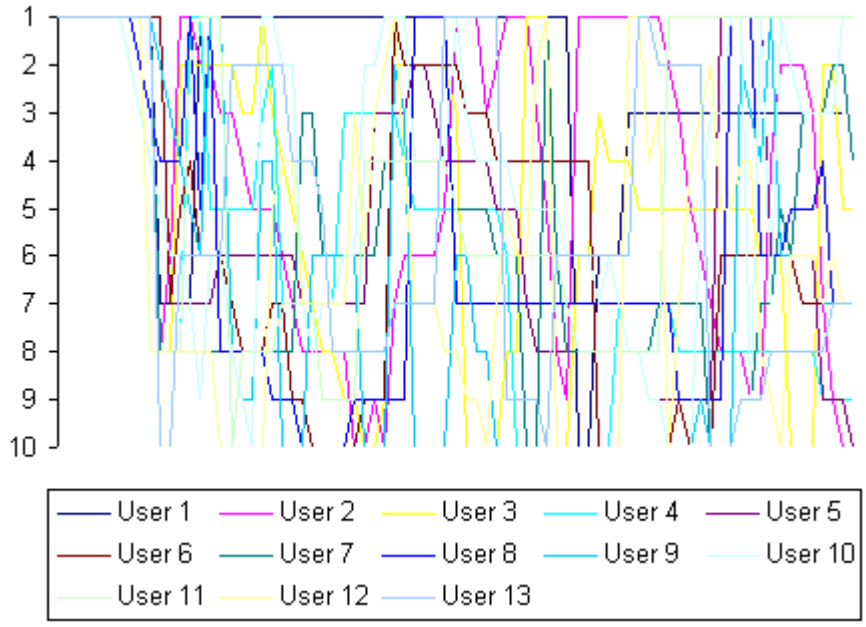

Fig. 24: Paths the Users took through Vision Europe

10.6.3 Mean Absolute Error

The Mean Absolute Error calculated for the two predictors induces that the Content-Based performs quite well, even though relatively little data is available. However, hybrid filtering does not perform that well. The Content-Based predictor performs significantly better than the hybrid predictor.[12]

[12] This has been proven by a T-Test with very high significance.

The MAE for forecast (i.e. the recommendation is calculated for the preceding asset as replacement for the actual asset to simulate the live context) and the non-forecast recommendations do not differ much (cf. Fig. 25).

Predictor	**Mean Absolute Error**
Content Based	0,71328
Content-Based (Forecast)	0,69881
Hybrid Filtering	0,96254
Hybrid Filtering (Forecast)	0,96413

Fig. 25: Overall Mean Absolute Error calculated for the different Predictors with and without forecast

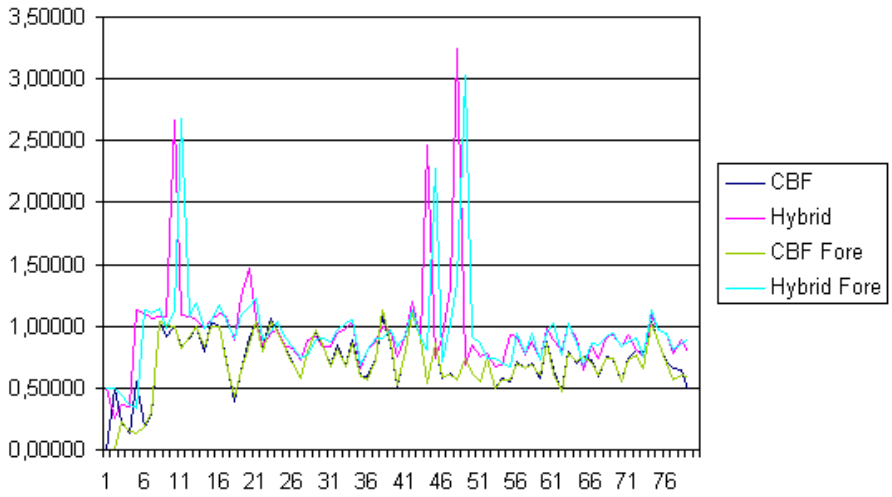

Fig. 26: Mean Absolute Error calculated for the different Predictors with and without forecast

Fig. 26 induces, that after a certain amount of time, the MAE stabilises around a value near the overall MAE. Both figures display the MAE calculated for the assets

at certain point in time. The MAE curves for the predictors with forecast resemble to those without.

10.6.4 Receiver Operating Characteristics

Receiver Operating Characteristics also support the assumption that the pure Content-Based predictor performs better in this case. As already proposed in chapter 10.6.1, ROC are calculated for the predictors with the two threshold values 0.5 and 1 (Fig. 27). Hence, the threshold value for ROC (and consequently also for being recommended) of 0.5 leaded to much better results than a threshold value of 1. This could be caused in the low usage of the “love and dismiss” feature.

Predictor	**Threshold**	**TPR**	**FPR**	**ACC**
Content-Based	0.5	0,68465	0,42105	0,65336
Content Based	1	0,11618	0,05592	0,36125
Content-Based (Forecast)	0.5	0,69571	0,47039	0,64654
Content-Based (Forecast)	1	0,15214	0,07895	0,37975
Hybrid	0.5	0,24205	0,11184	0,43330
Hybrid	1	0,00830	0,01974	0,29601
Hybrid (Forecast)	0.5	0,24758	0,13487	0,43038
Hybrid (Forecast)	1	0,00968	0,01645	0,29796

Fig. 27: Receiver Operating Characteristics for the different Predictors

The Content-Based predictor performs quite well with a threshold of 0.5 no matter whether forecasted or not. Content-Boosted Collaborative Filtering for example has a TPR of 0.6717 (Melville, Mooney & Nagarajan, 2001). In contrast, all other approaches leaded to very bad predictions, i.e. worse than random.

10.6.5 Interpretation

Both metrics induce that the Content-Based predictor performs better than the hybrid predictor. This might be caused in the bad base data, but the Content-Based predictor already performs quite well with this bad data. However, no final evaluation can be made unless more and better data is available after the 2008 field trial.

The worse performance of the hybrid predictor could be caused in the small number of test persons. As only 13 test persons were examined, not enough similar users are available for collaborative filtering. This assumption is also supported by the very different paths the users took through the interactive video mentioned in chapter 10.6.2.

Whether incorporating the time weighting factor provides better predictions than using pure predictions cannot be evaluated yet as all users in the evaluation are starting and finishing at the same time. This could be changed by just throwing away data of some test users. As already described in chapter 10.6.1, the behaviour at the beginning of the examination differed from the behaviour after a certain time, i.e. the watching behaviour tended to stabilise. Furthermore, the amount of available data is already very low so that no reliable statements could be given. Consequently, this should also be evaluated when the necessary data is available in mid 2008.

11 Discussion

Recommender systems have been an established means for more than a decade within interactive media research. They serve as basis for many commercial systems. Lately, the research focussed on real-time playlist creation like in last.fm or pandora.

This thesis presented a promising approach to incorporate recommender systems for interactive live media. The evaluation of the approach suffered from bad usage data. It was logistically impossible to provide good data, but that data will be gathered within a field trial in mid 2008.

One crucial aspect within live media was that it is impossible to provide appropriate content descriptions beforehand. Hence a substitute had to be found: every media asset in a channel is described by its predecessor, i.e. the recommender system computes recommendations as if the contents of the assets would not change. This assumption seems to provide sufficient asset descriptions.

Two approaches have been tested within this thesis: a pure Content-Based predictor and a hybrid prediction method. Though in most other cases a hybrid method performs better, this has not been the case here. Possibly, this is caused in the small number of test users so that finding similar users for collaborative filtering was very difficult. A comparison of both predictors should be carried out with the new data set.

Unfortunately, the time weighting factor could not be used as no learning curve of the Content-Based predictor could be generated. This factor might be an enhancement for the collaborative filtering part of the hybrid recommender.

The evaluation tests have shown a rather low usage of explicit feedback, i.e. favour and dismiss an item. It is not yet clear whether this was due to the fact that no means for giving explicit feedback was available within the GUI of the interactive DVD. Hence, GUI research about how a viewer could give explicit feedback in interactive television must be carried out.

The simple but powerful model of complex system for live recommendations used in the algorithm seemed to suit the requirements of interactive live television. It leads to high-quality recommendations based on low quality data. It can serve as a basis for further research in live recommender systems.

12 Outlook

As the algorithm already performs quite well based on really bad data, it should be retested with better data within the LIVE project. Additionally, a learning curve for the Content-Based predictor should be created. Hence, the time weighting factor could be calculated on that basis. Whether it leads to better predictions should be tested, too.

In order to be able to do this, the field trial should afford the necessary data. This data should contain usage statistics, i.e. when the user switched to which channel, and user ratings, i.e. favour and dismiss. Additionally, content descriptions for the assets should be available in some adequate format. Ideally, the field test would yield to a user data collection like MovieLens for live TV environments to be able to test different recommender algorithms.

Another aspect that should be researched is creating a means to calculate recommendations for a group of user instead of one individual. This approach should not be limited to a live context, but would also be interesting in other areas of recommender system research.

Furthermore, the relation of implicit and explicit feedback should be researched. As a lower threshold value for positive recommendation already leads to better predictions this would be a first approach. One other possibility would be incorporating a higher weighting of the explicit feedback. The test application already affords the necessary possibilities to do this.

Another important aspect would be an enhancement of the profile. This could be achieved by integrating new feedback methods like mood analysis or measurement of the viewer's environment. Furthermore, other profiling approaches, e.g. using questionnaires during registration, should be investigated.

Anymore, tagging of the media assets should be researched. Collaborative tagging approaches could be investigated and compared with work done by professional taggers. A hybrid approach combining both collaborative and professional tagging should also be considered.

References

ADOMAVICIUS, G. & TUZHILIN, A. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering,* **17** (6), pp. 734–749

AMAZON.COM. 2007. *Site Features: Recommendations.* [WWW] http://www.amazon.com/gp/help/customer/display.html/103-5031674-4340647?ie=UTF8&nodeId=13316081. (31-07-2007).

COSLEY, D. et al. 2003. Is seeing believing?: how recommender system interfaces affect users' opinions. *CHI '03 Extended Abstracts on Human Factors in Computing Systems (Ft. Lauderdale, Florida, USA, April 05 - 10, 2003).* New York, NY: ACM Press, pp. 585–592

FAWCETT, T. 2004. *ROC Graphs: Notes and Practical Considerations for Researchers. Machine Learning.*

FURNAS, G. W. et al. 1987. The Vocabulary Problem in Human-System Communication. *Communications of the ACM,* **30** (11), pp. 964–971

HALPIN, H., ROBU, V. & SHEPHERD, H. 2007. The complex dynamics of collaborative tagging. *Proceedings of the 16th international Conference on World Wide Web (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07.* New York, NY: ACM Press, pp. 211–220

JACOB, E. K. 2004. Classification and categorization: a difference that makes a difference. *Library Trends,* **52** (3), pp. 515–540

KLESKE, J. 2006. *Wissensarbeit mit Social Software. Konzeption & Entwicklung eines Systems für die kollaborative Wissensarbeit in der Forschung basierend auf Social Software* [Diplomarbeit]. Fachbereich Media - Media System Design. FACHHOCHSCHULE DARMSTADT. Darmstadt. [WWW] http://www.tautoko.info/JohannesKleske-Diplomarbeit-WissensarbeitSocialSoftware.pdf. (31-07-2007).

KÜBLBECK, C. 2007. Fröhlich, traurig, wütend oder erstaunt? *Mediendienst der Fraunhofer-Gesellschaft* (7).

LIVE CONSORTIUM. 2007. *LIVE - staging of media events.* [WWW] http://www.ist-live.org. (19-11-2007).

MARLOW, C. et al. 2006. HT06, tagging paper, taxonomy, Flickr, academic article, to read. *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia (Odense, Denmark, August 22 - 25, 2006). HYPERTEXT '06.* New York, NY: ACM Press, pp. 31–40

MCLAUGHLIN, M. R. & HERLOCKER, J. L. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (Sheffield, United Kingdom).* New York, NY: ACM Press, pp. 329–336

MECITV CONSORTIUM. 2004. *Media Collaboration for interactive TV.* [WWW] http://www.meci.tv. (24-10-2007).

MELVILLE, P., MOONEY, R. J. & NAGARAJAN, R. 2001. *Content-Boosted Collaborative Filtering.* Austin: University of Texas. [WWW] http://www.cs.utexas.edu/users/ml/papers/cbcf-sigir-wkshp-01.pdf.

MELVILLE, P., MOONEY, R. J. & NAGARAJAN, R. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. *In:* SCHULTZ, Alan C., ed.

Human-Robot Interaction: Papers from the 2002 Aaai Fall Symposium : November 15-17 North Falmouth, Massachusetts, Technical Report Fs-02-03. Menlo Park, CA: Association for the Advancement of Artificial Intelligence, pp. 187–192

O'DONOVAN, J. & SMYTH, B. 2005. Trust in recommender systems. *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces (San Diego, California, USA).* New York, NY: ACM Press, pp. 167–174

REDDY, S. & MASCIA, J. 2006. Lifetrak: music in tune with your life. *Proceedings of the 1st ACM international Workshop on Human-Centered Multimedia (Santa Barbara, California, USA, October 27 - 27, 2006). HCM '06.* New York, NY: ACM Press, pp. 25–34

ROSSON, M. Beth & CARROLL, J. M. 2002. *Usability engineering.* San Francisco, CA: Kaufmann. (The Morgan Kaufmann series in interactive technologies).

SCHACHTER, J. 2005. *del.icio.us: people who like recommendations also like…* [WWW] http://blog.del.icio.us/blog/2005/08/people_who_like.html. (31-07-2007).

WAGES, R. et al. 2007. Video Composer and Live Video Conductor: Future Professions for the Interactive Digital Broadcasting. *Journal of Virtual Reality and Broadcasting,* **4** (10).

WIKIPEDIA. 2007a. *Taxonomy.* [WWW] http://en.wikipedia.org/w/index.php?title=Taxonomy&oldid=147059878. (26-07-2007).

WIKIPEDIA. 2007b. *Folksonomy.* [WWW] http://en.wikipedia.org/w/index.php?title=Folksonomy&oldid=147191306. (27-07-2007).

WIKIPEDIA. 2007c. *Last.fm.* [WWW] http://en.wikipedia.org/w/index.php?title=Last.fm&oldid=149354594. (06-08-2007).

WIKIPEDIA. 2007d. *Receiver operating characteristic.* [WWW] http://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=155516298. (10-09-2007).

Appendix: Test Application

The test application is developed on Mac OS X Tiger and Ubuntu Linux 7.04 with python 2.4. Python is part of the standard installation of Tiger and Ubuntu Linux. It requires the libraries Qt3, PyQt3 and elementtree to be installed. All these libraries are also available on Windows, but Qt3 and PyQt3 are rather complicated to install on this platform. Qt3 and PyQt3 are only required for the profiler module. All other modules also work without these libraries as they are command line tools.

The application and the data gathered during profiling are included on the CD bundled with this thesis.

How to install Qt3 and PyQt on MacOS X

The easiest way to install Qt3 and PyQt3 is using the Linux package manager inspired MacPorts installer. MacPorts downloads, compiles and installs ports of Linux applications on a Mac.

1. Obtain and install the XCode development toolkit from Apple: http://developer.apple.com/tools/download/
2. Obtain and install MacPorts: http://www.macports.org/
3. Install the port py-pyqt3 using the following command in a terminal shell:
   ```
   $ sudo port install py-pyqt3
   ```

How to install Qt3 and PyQt on Ubuntu

The easiest way to install Qt3 and PyQt3 on Ubuntu is using the package manager apt on the shell:

```
$ sudo apt-get install python-qt3 libqt3-mt
```

How to install elementtree

Elementtree can be downloaded from http://effbot.org/zone/element-index.htm. Elementtree 1.2.6 is also included on the CD bundled with this thesis. To install elementree unpack it into a source directory, open a shell, change to that directory and type the following command in the shell:

```
$ python setup.py install
```

Possibly, you might need root rights for installation. In order to execute python with root rights precede the `sudo` command and enter your root password.

How to install the Test Application

The test application is included on the CD bundled with this thesis. To install the application, copy the contents of the directory `/test_app` to your hard disk. As the applications need write access to the application directory it cannot be started directly from CD. After that, change to this directory. All explanations bellow assume that this directory is your current working directory.

Profiler Application

The profiler application consists of three python modules: assetprofiler.py, actualassetprofiler.py and startprofiler.py. To start the profiler application run startprofiler.py:

```
$ python startprofiler.py
```

Data Management Application

The data management application consists of one python module: datamanager.py. It can be used to perform several data related tasks, e.g. export to csv and xml, delete tags and ratings etc. To start the profiler application run datamanager.py:

```
$ python datamanager.py <parameter>
```

You can select the performed action by adding one of the parameters specified below.

Parameter	**Action**
`-e <format>`	Exports data to XML and CSV files according to the parameter `<format>`. If format is omitted, the data will export to both formats. Otherwise you can use `xml` or `csv`.
`-del <what>`	Deletes data: The parameter `<what>` specifies which data is deleted: `tags` – all tags and tag associations, `ratings` – all ratings and `all` – all data
`-tag <channel> <newtag>`	Tags `<channel>` with `<newtag>`
`-deltag <oldtag>`	Deletes all occurrences of `<oldtag>`
`-reptag <oldtag> <newtag>`	Replaces all occurrences of `<oldtag>` with `<newtag>`
`-basetag <newtag>`	Adds `<newtag>` to tag list
`-rebuild`	Rebuilds tag list

`-fill <channel>`	Copies tags from asset 1 in `<channel>` to all assets in `<channel>`
`-backup`	Creates a backup of the data
`-restore <file> <what>`	Restores data from backup file `<file>`. What data is restored is specified in the parameter `<what>`. It can either be `tags`, `ratings` or `all`.
`-norm <length>`	Normalises ratings and tag associations to an overall movie length given by the parameter `<length>`

Fig. 28: Data management Application – Overview of the Command-line Parameters

Recommender Application

The recommender application consists of two python modules: calculatepredictions.py and pearson.py. To start the recommender application run calculatepredictions.py:

```
$ python calculateprediction.py
```

The recommender application creates four csv files containing the mean absolute error, and the Receiver Operating Characteristics for threshold value of 1and 0.5.

XML Schemas

The data management application provides an interface to export the data gathered by the profiler to XML. The following XML schemas are used to describe the XML files.

The file alltags.xml (cf. Fig. 29) stores the set of all tags $\mathfrak{T}$. Any tag used to describe an asset is mentioned here. The element `tags` holds the set of all tags. The name of the tags is stored in the attribute `value` of the element `tag`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="tag">
    <xs:complexType>
      <xs:attribute name="value" type="xs:NMTOKEN"
use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="tags">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tag" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Fig. 29: XML Schema for alltags.xml

The file assettags.xml (cf. Fig. 30) describes the assignment of tags to any asset. The root element `channels` holds the elements `channel`. The channel number is stored in the attribute `id`. A channel is built up from several elements of type `asset` identified by the attribute `id`. An asset has at least one element `tag`. The name of the tag is stored in the attribute `value`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="asset">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tag" maxOccurs="unbounded" />
```

```
      </xs:sequence>
      <xs:attribute name="id" type="xs:NMTOKEN"
use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="channel">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asset" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:NMTOKEN"
use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="channels">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="channel" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="tag">
    <xs:complexType>
      <xs:attribute name="value" type="xs:NMTOKEN"
use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Fig. 30: XML Schema for assettags.xml

The file ratings.xml (cf. Fig. 31) holds the ratings all users gave while watching the interactive DVD. The root element `ratings` holds the elements `user`. The user number is stored in the attribute `id`. A user has several elements of type `rating` identified by the attribute `asset` and `channel`. The rating the respective user gave

to that item is stored in the attribute `value`. Only assets that are rated with a non-zero value show up in this file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rating">
    <xs:complexType>
      <xs:attribute name="asset" type="xs:NMTOKEN"
use="required" />
      <xs:attribute name="channel" type="xs:NMTOKEN"
use="required" />
      <xs:attribute name="value" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="-1" />
            <xs:enumeration value="-2" />
            <xs:enumeration value="1" />
            <xs:enumeration value="2" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="ratings">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="user" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="user">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="rating" maxOccurs="unbounded" />
      </xs:sequence>
```

```
      <xs:attribute name="id" type="xs:NMTOKEN"
use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Fig. 31: XML Schema for ratings.xml

Appendix: Bundled CD

The CD contains the test application and this thesis as a PDF. If the CD is missing, it can be downloaded from http://martin-gude.de/go/master.

Appendix: BibTeX Entry

```
@mastersthesis{Gude:2007,
  author = {Gude, Martin},
  title = {Recommender Systems for Live-Transmission in
interactive Television: Gathering and Rating User Feedback},
  keywords = {recommender systems, interactive TV, live TV,
sports transmission}
  type = {Master's Thesis},
  year = {2007},
  address = {Cologne},
  school = {University of Applied Sciences Cologne},
  collaboration = {Gr{\"u}nvogel, Stefan M. Prof. Dr. and
Pla{\ss}mann, Gerhard Prof. Dr.},
  url = {http://martin-gude.de/go/master}
}
```

Acknowledgements

I would like to thank the following people who supported my thesis in the last months: Dr. Andreas Pütz, Prof. Dr. Stefan Grünvogel, Prof. Dr. Gerhard Plaßmann, Julia Hoffmann, Roland Westermaier, Dr. Barbara Schinzel, Frank Potthof, Torsten Kliemand, Dirk Krause, Gero Grebe, Jörg Jung, Roland Gude, Malte Donay, Björn Starr, Frauke Voss, Richard Wages and my colleagues at Pixelpark Agentur Cologne.

Jurat

For legal reasons the following jurat is in German language. The English translation has no legal consequences. Only the German original is applicable.

“I assure, that I have written this thesis independently. I marked all portions that I have taken literally or in general manner out of published or non-published works as quotations. All resources and aids, which I used for the work, are indicated. The work has not been presented with same contents or in substantial parts to any other test authority.”

Eidesstattliche Versicherung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Cologne, 30 November 2007