Telecommunications engineering

DESIGN OF AN EXPLANATION ENGINE FOR RECOMMENDER SYSTEMS

Diploma thesis project

Author
Iván Blanco Guerrero

Supervisors:
Dr. Thomas Kemp, Sony
Javier Alonso García, Sony
Dr. Beatriz Otero Calviño, UPC

February 2013
Aim for the moon. If you miss, you may hit a star.
W. Clement Stone
Acknowledgements

I did my master thesis in Sony Deutschland GmbH in Stuttgart. During this time I gathered a lot of knowledge in different areas but most importantly in recommendation systems and machine learning algorithms.

I am very grateful to all the people at EuTec, Speech and Sound Group. Especially, I would like to acknowledge Dr.Thomas Kemp and Mr.Javier Alonso for giving me the opportunity of doing this master’s thesis. They have provided me ideas and advices during the whole time. Without their support and trust, the result would not have been so successful. I also thank Dr.Beatriz Otero from for their ideas and helping me and review my work constantly.

I would also like to thank all my friends from Barcelona, who have accompanied me all over these years, and the ones I have met these last few months in Sony while working on this thesis. Without them, the stay abroad would become much harder and boring. Finally, a very special thanks to all my family, for their unconditional support and encouragement.

Iván Blanco
February, 2013 in Stuttgart
Abstract

This report is an overview of the area of explanations in recommender systems. A recommender system has the ability to establish trust and convince the users with its recommendations. This part is a very important factor especially for e-commerce environments. Another important aspect is the ability to explain the recommended results which is the best way to establish/build a trust model for recommender agents and give to the user the perception of high competence.

Recommender Systems serve online customers in identifying those items from a variety of choices that best match their needs and preferences. In this context explanations summarize the reasons why a specific item is proposed and strongly increase the user’s trust in the system’s results.

The identification of the best-fitting products is in many cases a complex decision making task which forces users to fall back to different types of decision heuristics.

I present in this thesis the results of my work in developing an explanation engine for a commercial recommender system and a different way to build an explanation interface.

Keywords: Recommender systems; Explanation interfaces; Trust model; Interface design.
## Contents

1 **Introduction** 3
   1.1 Motivation ............................................. 3
   1.2 Problem statement .................................... 4
   1.3 Research objective ................................... 5

2 **Quick overview of Recommender Systems** 7
   2.1 Recommender techniques ............................... 7
   2.2 Collaborative Filtering (CF) ........................... 8
      2.2.1 Singular Value Decomposition (SVD) ............. 9
      2.2.2 Restricted Boltzmann Machines (RBM) .......... 11
   2.3 Content-based Filtering (CBF) ......................... 13
   2.4 Knowledge-based (KB) ................................ 15
   2.5 Metrics to evaluate recommender systems............... 16
      2.5.1 Accuracy metrics ................................. 16
      2.5.2 Classification metrics ........................... 16
   2.6 Recommender systems issues ............................ 17

3 **Explanations Systems** 19
   3.1 State of the art ........................................ 19
      3.1.1 Designing approach ............................... 20
   3.2 Impact of Explanations ................................ 22
      3.2.1 Main benefits and goals ........................... 22
      3.2.2 Other goals ....................................... 26
   3.3 Explanation models ..................................... 27
      3.3.1 Demographic ....................................... 27
      3.3.2 Collaborative filtering explanations ............... 27
      3.3.3 Content based explanations, the "Why" style ........ 29
      3.3.4 Knowledge-based explanations, the Trade-off style .... 31
      3.3.5 Hybrid Critiquing-based Recommender Systems ....... 35
   3.4 Evaluating the impact of explanations .................. 36
      3.4.1 Subjective evaluation .............................. 36
      3.4.2 Mathematical evaluation ........................... 37
   3.5 Design guidelines ....................................... 39
   3.6 Other aspects ........................................... 40
Chapter 1

Introduction

1.1 Motivation

Have you ever played a game that somebody recommended to you, and felt you wasted your time? If yes, for sure you have thought what was the reason behind it and most importantly, why? To prevent this problem, an argumentation about this recommendation may be the solution.

Before Internet, a customer had limited access to the information regarding the product itself as well as other possible options. Advertising was mainly the only way to promote products and the problem for the user was how to contrast this information. In the case of cultural products such as music or books, specialized magazines played the role of broadcasters of presenting what is new. Nowadays the situation has been completely reversed. Due to the increasing size and complexity of product assortments customers are challenged to find the items best fitting their wishes and needs. We have passed the scarcity of information to saturation. We have passed to see some sections in our shopping centres to have access to an unlimited number of web content in on-line shops or sharing networks. Now the problem is how to separate between what we want to find and what we do not want to find.

To deal with this problem, recommender systems try to predict the rating or preference that a user would give to an item that he had not yet considered. These systems are included in the Machine learning field and have become extremely common in the last few years in on-line sales. Examples of such recommenders are Amazon for books and other items, Netflix for movies, youtube for videos even Facebook for friends. Moreover, some vendors have incorporated recommendation capabilities into their commerce servers such as apple or Linked-in.

While most researches in the field of recommender systems have focused on improving the accuracy of recommendations, recent work suggests another set of goals that can be achieved with argued recommendations. These goals can
be: trust, transparency, satisfaction or even something deeper as loyalty to the system or the intention to come back to get more recommendations. The key to achieve this set of goals is to explain this recommendations. While recommendations tell users which items they might like, explanations reveal why they might like them.

Explanations help users make more accurate decisions, improve user acceptance on recommendations, and increase trust in the recommender system. Moreover, studies\textsuperscript{1} indicate that 86% users want explanations of their recommendations. Being able to effectively explain results is also essential for product recommender systems. Explanations in recommender systems can be generally understood as a form of communication between a selling agent (the recommender systems ) and the on-line customer. Thus, explanations make the reasoning process of the recommender systems transparent to the users and help them to comprehend why an item is deemed to be relevant.

1.2 Problem statement

The current generation of recommender systems still requires further improvements to make recommendation methods more effective and applicable to a broader range of real-life applications. However, to make predictions about the tastes of the users, we need to manage large amounts of data, sometimes Terabytes of information. Purchases matrix is large and the calculation of the similarity matrix between items/users is difficult. This requires an extreme computational cost which sometimes cannot be assumed.

An example of this problem is The Netflix Prize and the Recommendation Problem\textsuperscript{1}. In 2006 Netflix announced a data mining competition for movie rating prediction. They offered a $1 million prize to whoever improve the accuracy of the existing system by 10 %. The race was on to beat the Netflix Root Mean Square Error (RMSE) of 0.9525 with the finish line of reducing it to 0.8572 or less. After a year into the competition, the Korbell team (AT&T) won the first Progress Prize with an 8.43% improvement. The team reported more than 2000 hours of work and the final combination of 107 algorithms. They highlighted two algorithms with the best performance in the ensemble: Matrix Factorization (which the community generally called SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM). A linear blend of these two reduced the error to 0.88. But it was not enough. On 2009, the team "BellKor's Pragmatic Chaos" achieved a 10.05% improvement over the previous system and winning the prize with a RMSE of 0.8567. Although this was a great improvement, it was never implemented due to some limitations, for instance the winning algorithm was designed to handle 100 million ratings, instead of the more than 5 billion that Netflix had. Furthermore another problem is that the algorithms were not built to adapt to members who added more ratings, so it

\textsuperscript{1}http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html
suffered scalability problems. The Netflix team evaluated the code of the new methods but the additional accuracy gains that they measured did not seem to justify the engineering effort needed to bring them into a production environment.

Nowadays, it seems that recommendation algorithms have reached an accuracy limit. For this reason other ways to improve the user experience in recommendations are being investigated. And the industry behind recommendations insisted on the significance and the high importance of the explanations for the recommendations, as they can compensate the lack of accuracy and achieve other objectives.

1.3 Research objective

The purpose of this thesis is the design and implementation of an Explanation Engine to improve the current PlayStation Store® Recommender System, incorporating personalization and argumentation to these recommendations.

To do this, it has been necessary to make a preliminary study about the current explanation systems analysing its advantages and drawbacks to solve initial problems and limitations. To our knowledge there is no existing work on explanations for game recommender systems. After the study we decided to implement an Explanation Engine for the PlayStation Store® with following requirements:

1. To explain the given recommendation to the user
2. The explanation must be personalized, i.e., tailored to the users tastes or demographic information
3. The system should also provide different tailored alternatives to the proposed recommendations, with their corresponding explanation
4. In order to analyse the usefulness of the explanations, an evaluation system will be developed. The evaluation criteria for this system need to be defined and adjusted to the explanation system, and they will consist of both objective measures (e.g. Coverage etc) as well as subjective measures (e.g. survey results).

With this work it is also wanted to provide a unified view on the different recommendations paradigms and the explanation ways, and lay the groundwork for future implementations.

---

2By the time this work ended, Microsoft introduced a recommender system to their Xbox Live Marketplace in RECSYS 2012 - Conference on Recommender Systems, but without any argumentation for its recommendations.
In this thesis, several ways will be described to extend the recommender systems capabilities using argumentations. However, before doing this, a quick overview about recommender systems is presented in section 2. Later in section 3 a survey of the state of the art in explanations systems is exposed. Then the implementation of a prototype and its results are explained in section 4 and 5.
Chapter 2

Quick overview of Recommender Systems

2.1 Recommender techniques

A recommender system has the goal of identifying a set of products fitting the wishes and needs of a customer. To do this, an underlying algorithm is needed. In this chapter, a review of the existing research work in recommender systems will be presented, explaining the metrics of evaluating them and finally commenting on some of their issues.

The algorithms can be classified in three big families: Collaborative Filtering (CF), Content Based Filtering (CBF) and Knowledge Base (KB).

Figure 2.1: Archetypes of domain
a) Pure CF, b) SVD/RBM, c) CB, d) KB
2.2 Collaborative Filtering (CF)

Collaborative filtering is a method of making predictions about the interests of a user by collecting preferences from many users. It is one of the most well known recommendation approaches. Basically it simulates the social process of asking a friend for a recommendation; that is, if Adela and Melanie have similar tastes, then let us recommend to Adela the movies that Melanie likes and she is not aware of.

This algorithm was popularized by Amazon.com’s recommender system. "People who buy X also buy Y...". The advantage of the collaborative filtering approach is that it does not rely on metadata content, only on rating data, and therefore it is capable of accurately recommending complex items without requiring or "understanding" the features of the item itself.

Notice that there are two variants of this algorithm: User based algorithm (CF2u) and Item based algorithm (CF2i). They are symmetric in the sense that only the input (user/item) is inverted, but the algorithms are the same.

To generate a prediction we need to do some previous steps. The preference relations between users and items are encoded by a rating matrix $R$ where the rating of a user $u$ over a item $i$ is stored in the element $R(u,i)$. An example of this rating matrix is given in Figure 2.2.

![Figure 2.2: Collaborative Filtering scheme prediction](image)

In the next phase the similarity matrix is calculated as a product of independent factors:

\[
\text{Similarity} = \text{Matching Factor} \times \text{Recall Factor} \times \text{Regularization Factor} \quad (2.1)
\]

Where:

Matching Factor: This terms refers to the similarity computation for the co-ratings. It is a mathematical way to compare multidimensional vectors. Plain Cosine, Pearson Correlation Jacquard Coefficient and Tanimoto coefficient are
CHAPTER 2. QUICK OVERVIEW OF RECOMMENDER SYSTEMS

some of the most used.\(^1\)

Recall Factor: This is a normalization term that tries to take into account the size of the co-ratings set.

Regularization Factor: In case we do not have many ratings for both items, we approach the global similarity instead of assigning a zero. No information does not mean not similar, only unknown.

After this calculation we get a Similarity Matrix with the dimensions: \((total\,Num\,Items \times total\,Number\,Items)\). This matrix is symmetric and has an all 1’s diagonal. These similarity relationships are exploited to generate predictions to recommend items that are either similar to the items a user has liked in the past or that are liked by similar users. The next step is to predict the rating for the prediction of the user-item pair \((u, i)\) as:

\[
\text{Prediction} = \frac{v_I \times s_I^T}{\sum s_I} \quad (2.2)
\]

Where \(v_I\) is a vector with all the ratings for item \(i\), and \(s_I\) is a vector with the corresponding similarities for all bigger than 0. After the prediction calculation, it is recommended to apply some factors to improve the accuracy of the recommender. For instance Novelty Degree or removing the Global effects.

- **Advantages and Drawbacks of CF**
  - ✓ Very accurate rating predictions.
  - ✓ No need to know the item attributes to recommend it

  × For a new item added, the collaborative filtering system would not be able to recommend it until it is rated by a substantial number of users. It means less novel recommendations

  × Due to the data sparsity, new users will need to rate sufficient number of items to enable the system’s recommendations. See Cold start problem in section 2.6

  × It suffers serious scalability problems due to the high number of users or items.

  × It could be attacked externally by people who give a lot of positive ratings for their own/favourite items and negative ratings for their competitors. See 2.6

2.2.1 **Singular Value Decomposition (SVD)**

Is a specific variant of collaborative filtering, this technique uses data mining and machine learning algorithms to find patterns in the training data. These

\(^1\)For more details see [2].
patterns are then used to make predictions on the new data.

It has traditionally been used for Information Retrieval in documents and terms. In the proposed recommender scenario we do not have all the information, 1 means "purchased" and 0 means not purchased yet. For this reason, the input matrix in this case is very sparse.

The basic model identifies $k$ abstract factors that can explain most of the signals in the ratings. However, abstract properties rarely capture semantic meaning that can easily be communicated to users.

This technique tries to build user and item profiles like this:

$$R = U \cdot S \cdot V'$$

Where Matrix $S$ is a diagonal matrix containing the $r$ singular values of matrix $X$. If we keep only the biggest $k$ singular values in $S$, this will give us the best rank-$k$ approximation to $R$. The problem with this algorithm is that we need to estimate the optimum value of $k$ to get the best Precision-Recall curve (see section 2.5).

The prediction of user $i$ for product $j$ is calculated by:

$$\text{Pred}(i, j) = r_i + U_k \sqrt{S_k(i)} \sqrt{S_k V'_k(j)}$$

(2.3)

It is generally very interesting to combine SVD with other algorithms such as CF. Indeed, SVD-matrix factorization works at a regional scale whereas neighbourhood based algorithms work at a local scale. The idea is first to apply an appropriate space transformation (remove noise, reduce dimensionality) and then apply memory-based algorithms. It is called SVDu the combination of SVD with CF2u and SVDi when combined with CF2i.

$$R_{red} = U_k \cdot S_k \cdot V'_k$$

Figure 2.3: Example of Singular Value Decomposition

- **Advantages and Drawbacks of SVD**
  - ✓ Very accurate rating predictions.
  - ✓ Removes the noise hence improves the accuracy.
  - ✓ Reduce dimensionality, therefore is faster and memory friendly
  - ✗ There seems to be an optimum parameter $k$ for each database.
  - ✗ Difficult semantic meaning.
  - ✗ Each time logs are added, the matrix must be decomposed again.
2.2.2 Restricted Boltzmann Machines (RBM)

Restricted Boltzmann Machines is a neural network based algorithm and it is used for solving the rating-prediction problem. It essentially performs a binary version of factor analysis. Instead of users rating a set of movies on a continuous scale, they simply tell you whether they like a movie or not, and the RBM will try to discover latent factors that can explain the activation of these movie choices.

It works like stochastic neural network, where each unit has binary activations and depend on the neighbour they’re connected to. Stochastic meaning these activations have a probabilistic element. These activations have a probabilistic element consisting of:

- One layer of visible units (for example users’ movie preferences whose states we know and set)
- One layer of hidden units (the latent factors we try to learn)
- A bias unit (whose state is always on, and is a way of adjusting for the different inherent popularities of each movie)

For example, suppose we have a set of six movies (Harry Potter, Avatar, Lord Of The Rings 3 (LOTR3), Gladiator, Titanic, and Glitter) and we ask users to tell us which ones they want to watch. To do the recommendations we need to discover which latent units are underlying these movies. For instance, two natural groups in our set of six movies can be done, the first one SF/fantasy (containing Harry Potter, Avatar, and LOTR 3) and the second one Oscar winners (containing LOTR 3, Gladiator, and Titanic), so the goal is that our latent units will correspond to these categories. Then our RBM would look like the following:
Later we can "ask" RBM which of the hidden units the user activates with his preferences and generate a set of recommendations. For a more accurate and mathematical description see Adela’s thesis[19].

- **Advantages and Drawbacks of RBM**
  - **✓** Very accurate rating predictions combined with other algorithms (SVD or CF2i)
  - **✗** Difficult to implement with several types of data.
2.3 Content-based Filtering (CBF)

Content-Based filtering systems make recommendations for a target user based on the past data of that user without involving data from other users. It is based only on content data. Recommendations are computed by determining those items that are most similar to items the user is already known to like. In other words, it is based on exploiting similarities between user preferences represented by purchased products and existing product descriptions.

We assume matrix \( F \), where \( F(i, f) \) means that item \( i \) contains feature \( f \) and zero otherwise. For each item we have several features that describe its characteristics. For the construction of the feature profile of a user, it is necessary to define a positive rating threshold \( P_T \), to select the items whose rating express a positive preference by the user.

![Feature Matrix](image)

Figure 2.5: Item-Feature Matrix and User-Feature Matrix for Collaborative Filtering.

To calculate the similarity matrix, first of all each feature of meta-data is considered independently and the needed weights for each category is provided.

\[
\text{Similarity} = \sum_{f_{\text{feat}}=1}^{k} W_{f_{\text{feat}}} \cdot \text{Similarity}_{f_{\text{feat}}} \quad (2.4)
\]

Later the predictions are computed. The traditional approach consists of taking the neighbours list associated to the position \((u, i)\) for which we want a prediction, and then calculating it as the sum of contributions from each neighbour and his position. This technique works well with Forced Recommendations.\(^3\)

\[
\text{Prediction} = \sum_{j=1}^{N} (\text{Similarity}_{ij} \cdot f(\text{pos}_j)) \quad (2.5)
\]

\(^3\)The system predicts all the \((u, i)\) relationships contained in the Test Set
This technique is based on the meta-data of both users and items. There are three approaches to apply this kind of filtering:

1. Item based algorithm (CBFi)- Calculates item similarities by directly comparing its metadata.

2. User based algorithm (CBFu)- Builds a user profile, which is a metadata collection of all items ranked by that user. Later it compares his profile with other users with similar profiles.

3. User-Item profile algorithm (CBFp)- calculates similarities by comparing directly the meta data that is available for users and items. It is a mixture between CBFu and CBFi.

CBF gives generally worse accuracy results compared to CF, and suffers from the blinkers defect: since you recommend items with very similar content, variety is impoverished. On the other hand, CBF can recommend less popular items (less rated/purchased items), and allows novel/surprising recommendations. Even if CF algorithms are known to be more accurate, they suffer from sparsity and scalability problems, which can be solved by reducing dimensionality first. Another important aspect is the latency problem is that items with few ratings are undervalued, so it takes some time to start recommending new items. CF might also be inappropriate for Grey-sheep users (users with unusual preferences). As a result, CBF and CF should be combined in order to avoid all these problems.

- **Advantages and Drawbacks of CB**
  - ✓ Easy to explain, because it explain similarities between items by disclosing their property relationships.
  - ✓ High diversity.
  - ✗ Worse accuracy results compared to CF.
  - ✗ Blinkers defect
2.4 Knowledge-based (KB)

Knowledge based algorithms are characterized by additional domain properties such as abstract users requirements or preferences as well as various relationships between them. They, for instance, encode explicit sales expertise like which item features help to fulfil a specific user requirement.

The solutions are computed by identifying those items that satisfy all the domain restrictions. In case of no solution, the user is either asked to revise his preferences or the reasoner relaxes some of the domain restrictions. The historic MYCIN expert system also included this feature, only disclosing rules which actually contributed to an answer.

In most cases users do not have a clear view of their preferences and therefore construct preferences in the course of a recommendation session. Consequently, the design of a recommender application can influence the outcome of the decision making task.

Thus this technique is able to support intelligent explanations and product recommendations which are determined by a set of explicitly defined constraints. In contrast to collaborative and content-based filtering, knowledge-based approach is in the majority of cases applied for recommending complex products and services such as consumer goods, technical equipment, or financial services.

This algorithm itself is an explanation form, it is a kind of hybrid between recommender and recommender+explanation systems. So It will be discussed further in the next chapter.

![Figure 2.6: Knowledge base scheme](image)

---

MYCIN, 1972, was an early expert system that used artificial intelligence to identify bacteria causing severe infections, such as bacteremia and meningitis, and to recommend antibiotics, with the dosage adjusted for patient’s body weight. Research conducted found that MYCIN had an approximate correctness rate of 65%, which is better than the majority of physicians who are not specialists in diagnosing infections, and only slightly worse than physicians who were experts in that field (who had an average correctness of approximately 80%).
2.5 Metrics to evaluate recommender systems

In the context of recommender systems the metrics normally used are Predictive and Classification accuracy metrics. They provide a factor with the quality of our recommendations.

2.5.1 Accuracy metrics

**Root Mean Squared Error** (Root Mean Squared Deviation, RMSD) measures the difference between the values predicted by a model and the actual values. It is a measure of accuracy. The lower RMSE we get, the better predictions are.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (C_i - \bar{C}_i)^2} \quad (2.6)
\]

There are other tools like:
- RMSE@X Same performance measures but only for the top X recommendations.
- ErrorRate@X Given the X first recommendations, this is the number of non purchased items. !!!!!!!!!!!!!!!!!!Review!!!!

2.5.2 Classification metrics

- **Precision**, measures the precision of the recommendations made by recommender systems and it is measured by the quantity of recommended items that are actually interesting to the user in comparison with the set of all recommended items.

\[
Precision = \frac{|\text{(relevant items)} \cap \text{(recommended items)}|}{|\text{(recommended items)}|} \quad (2.7)
\]

Figure 2.7: The precision of a system shows how close the prediction is to the actual evaluation done by the user.
• **Recall**, indicates the quantity of interesting items to the user that appear in the recommendation list in comparison with all relevant items.

Recall = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|} \quad (2.8)

Figure 2.8: It gives us how many of the subset of all the liked items are recommended.

### 2.6 Recommender systems issues

**Cold Start** The problem occurs at the start of the system (no assessments from other users). A recommender system usually compares the user profile with some reference characteristics. These characteristics can be based on information (content-based approach) or on the user’s social environment (collaborative filtering approach). In the content-based approach, the system should be able to match the characteristics of an item to relevant characteristics in the user’s profile. In order to do that, a model with sufficient detailed information on the user, including his tastes and preferences must first be built. This can be done explicitly (by consulting the user) or implicitly (observing the user’s behaviour). In both cases, the Cold Start issue requires the user to create his/her profile before the system can begin any relevant recommendation. Because of the Cold Start issue, items not previously assessed would be ignored in the collaborative filtering approach.

**Gray Sheep** If a user has a specific taste, the recommendation may not be accurate, as there are no "close neighbors". This problem is called gray sheep (Resnick and Varian 1997) (Lorenzi 2006). In the collaborative filtering system, a user with this profile is not easily related to other users in the system, making it difficult to recommend items. In the content-based filtering system, even if the user has a rare profile, the recommendation of items related to this profile is not an issue, since recommendations are more generic. For example, if the system identifies that a user is interested in technology and oceanography, it will easily recommend these items to the user, even if only unpopular items have been evaluated.
Early-Rater When a new item emerges, it cannot be recommended to a user before a person assesses it (Resnick and Varian 1997) (Lorenzi 2006). This issue is clearly identified in collaborative filtering. When a new item with no user assessment or recommendations is inserted, it cannot be recommended. In content-based filtering, knowing the contents of an item is enough to enable a recommendation to a user.

Sparse evaluations When there are few users and many items, the evaluations may become sparse and it becomes difficult to find similar users. In collaborative filtering, this issue is easily identified because the filtering is completely based on the user’s assessment of the item. In content-based filtering, the recommendation does not depend on the number of users and items, but rather on their profiles and contents.

Specialization Only items that are similar to those previously evaluated by the user will be recommended. In content-based filtering, this issue is clearly identified. A user whose profile has been defined will always receive items related to his profile, and any personal profile modification (outside the system) will not be reflected on the system. In collaborative filtering, item recommendation is not based on the user’s initial profile, but rather on his/her actions and relation to other users.

Serendipity This is related to the lack of surprise in the recommendation. Products that are not related to the user’s profile may never be recommended. This problem occurs in content-based filtering, since the recommended content will always belong to the same group relating back to the user profile. Meanwhile, in collaborative filtering, the surprise occurs more frequently, since similar users may have evaluated completely different items from those seen by the original user.

Scalability When the quantity of users, items and evaluations is too large, the system that executes real-time calculations of the relations among users may provide a very long response time and may need computer resources that are not available. This is a common problem in both approaches. However, in collaborative filtering, this issue is more evident as the calculations are done using all the users and all the items. In content-based filtering, calculations are done using only one user and all related items, considering all attributes.

External attacks An external attack against a collaborative filtering recommender system consists of a set of profiles, each contains biased rating data associated with a fictitious user identity, and including a target item, the item that the attacker wishes the system to recommend more highly (a push attack), or wishes to prevent the system from recommending (a nuke attack). This attacks don’t affect large systems due to they have high quantity of data.
Chapter 3

Explanations Systems

3.1 State of the art

The identification of the best-fitting products is in many cases a complex decision making task which sometimes forces users to fall back to different types of decision heuristics. But at the end of this task the user must decide between the collection of proposed items. To help with this decision an argumentation can be provided by the system. In general, an explanation is a set of arguments to describe a certain aspect, an item or a situation. An argument is a statement containing a piece of information related to the aspect which should be explained, "The gas station is inexpensive" or "The game is popular".

Typical explanations for product recommendations include constructions such as "the digital camera Cyber-Shot is well suited to your needs because you would like to take pictures of your children playing football" or "it is a lightweight compact camera especially designed for action photos", or in the movie domain "the film Avatar was extremely well received by science fiction fans and so you will probably enjoy it too". Such information is commonly exchanged between a sales assistant and a customer during in-store recommendation processes and is usually called an explanation. Text, images, video and combinations thereof can be used to explain a system’s output and can be either explicitly requested by users or automatically displayed by the system. Even they can be positive, negative or neutral depending on the designers intention.

![Figure 3.1: An explanation is additional information to explain the system’s output following some objectives](image-url)
But are explanations really needed? Studies indicate that the 86% users want explanations for the recommendations. Because, they may not accept the recommendations if they do not understand why something was recommended to them. Also Recommendations without control can lack user acceptance can cause the users to mistrust the system. Furthermore, industry related to this field after the Netflix problem is researching how to incorporate explanations to their systems. An example of this was the ACM Recommender Systems 2012. They insisted on the significance of the user interface and the high importance of the explanations.

Regarding research on explanations, MYCIN was the first expert system incorporating a primary version of explanations. Years passed, and many pure CB systems have tried to provide explanations to users. For instance, Billsus and Pazzani in their work, recommend news articles to users, providing also explanations for reasoning their recommendations. In 2000, Mooney and Roy proposed a method based also on pure CB for recommending books. These works were pioneering for the problem of explanation and inspired subsequent research on combining CF and CB for explanation purposes. In the area of CF, there is a little existing research on explanations. In 2000, Herlocker proposed 21 different interfaces of explaining CF recommendations (See Annex). By conducting a survey, they claim that the "nearest neighbour" style is effective in supporting explanations. Amazon.com's recommender system early adopted the "nearest neighbour" explanation style. In 2005, Bilgic demonstrated through a survey, that the "influence" and "keyword" styles are better than the "nearest neighbour" style. This is because they help users to accurately predict their true opinion of a recommendation.

3.1.1 Designing approach

To design explanations for recommender systems, some authors [3] categorize different approaches for explaining recommendations based on design principles or the impact on their users:

1. Design principles
   There are all the factors that determine the generation of explanations for recommender systems. Structural characteristics such as length, writing style or confidence could be some examples. The major design principles are detailed in figure 3.2.

2. Impact on users
   Depending on the desired benefit, explanations strategies may change to achieve some goals. This part is extended in the next section.

If we try to sum up all the Information categories that can be used for generating explanations we can differentiate between three different aspects of input:
CHAPTER 3. EXPLANATIONS SYSTEMS

Figure 3.2: Dimensions for categorizing explanation approaches

- **User model**: The explanations are tailored to the system’s beliefs about the given user. For example, the system could present arguments based on the user’s known ratings, preferences or demographics.

- **Recommended item**: The explanation is dependent on the specific recommended item. For example the explanations make statements about the specific characteristics of the recommended item.

- **Alternatives / trade-off**: The explanations argue in favour of or against alternatives to the recommended item.

Obviously, explanations can be tailored to all three combinations as shown in Figure 3.3. This could be useful to put all the different kinds of explanation in order and try to achieve the most complete explanation. However, in most cases only one axis is exploited when doing recommendations.

Figure 3.3: Information categories
It should be noted that several explanations can exist for a recommendation. In addition, systems generating explanations must consider the presented solutions to avoid spurious explanations. This minimal knowledge base serves as a basis for generating explanations, but is not enough on its own. This is because explanations are the inputs for building arguments for humans. Some parts of the knowledge base may be necessary for entailment but are obvious for humans. For instance, classifying a van with 7 seats as a family car is obvious for most users whereas a recommender system has to deduce this classification from data sheets in cases where this is not given.

3.2 Impact of Explanations

As said before, incorporate explanations to the recommendations give the system extra features to compensate the lack of accuracy. In this chapter all this benefits will be summed up.

If we try to synthesise explanation taxonomy, we can define explanations in recommender systems by two properties:

- Information about the recommended item
- Underlying benefits and goals

This second point is the most important and the most difficult to achieve correctly.

3.2.1 Main benefits and goals

Explanations may help isolate and correct misguided assumptions or steps about user decision. In [4], seven generalizable goals for explanations in recommender systems are provided. In this section some of them will be explained. It should be noted that the goals applicable to the single item recommendations are the only ones mentioned. For example when a single recommendation is offered (in our context, video-games). When recommendations are made for multiple items, such as a list, the criteria may be different and other factors such as diversity are considered.

The table 3.1 summarizes these goals. They are similar to those desired in expert systems.
CHAPTER 3. EXPLANATIONS SYSTEMS

<table>
<thead>
<tr>
<th>Aim</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparency</td>
<td>Explain how the system works</td>
</tr>
<tr>
<td>Scrutability</td>
<td>Allow users to tell the system it is wrong</td>
</tr>
<tr>
<td>Trust</td>
<td>Increase users’ confidence in the system</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>Help users make good decisions</td>
</tr>
<tr>
<td>Persuasiveness</td>
<td>Convince users to try or buy</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Help users make decisions faster</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>Increase the ease of use or enjoyment</td>
</tr>
</tbody>
</table>

Table 3.1: Explanatory criteria and their definitions

It is important to separate these goals because all of can be seen as trade-off. For instance, while personalized explanations may lead to greater user satisfaction, they do not necessarily increase effectiveness. Transparency does not necessarily aid trust. They are to be constructed as isolated benefits which in turn are related to others.

The type of explanation that is given to a user depends on the criteria of the recommender system designer. For example if the system sells books, the main criteria could be the user trust. But if the criteria is for selecting TV-shows, user satisfaction could be more important than effectiveness. But explanations should be part of a cycle where the designer proposes a goal and the user understands what is going on in the system and exerts control over the type of recommendations made.

If users could comprehend which information was used to create the recommendations for them, they will understand that the adaptation in the system was based on the personal attributes stored in their profile. At this point, they will discover they can control the personalization by the information provided by them on their profiles. This control can reinforce the primary idea of the designer. Now, the main characteristics of these goals will be explained.

Transparency

Transparency should give an honest account of how the recommendations are selected and how the system works. It should try to answer the user’s question: Why was this recommended to me? And provide the necessary information as to why one item was preferred over another. When a recommender system outputs a solution, a client might ask for a justification for the proposed solution. Such explanations are called how-explanations because they exploit information on how a conclusion was deduced or how a sequence of rules was activated in the decision making process. The implicit assumption was that the model employed to give recommendations can also serve as a model for the argumentation that the advice is plausible from the client’s viewpoint[3].
Sometimes these deductions are not always accepted as high quality explanations, typically for two reasons. First, depending on the underlying algorithm, the deduction traces may be far too complex and may confuse the client, for example in matrix factorization algorithms (SVD). Second, the rules exploited for deduction are not necessarily accepted as valid arguments for approving a solution because they only shallowly reflect the known principles and laws of the underlying algorithm. For instance, collaborative approaches exploit the similarities between users and items, Knowledge-based recommenders exploit known dependencies between properties of items and users. And the mistake could be "... someone with a large family and income will prefer large houses".

Other authors (Herlocker, Konstan, and Riedl, 2000) differentiate between white box and black box explanations. White box discloses and exploits the underlying model of the recommendation engine, while black box explanations do not disclose the operation of the system to the user.

**Scrutability**

This characteristic is present when the system allows to user to correct reasoning and suit their preferences. While scrutability is very closely tied to the criteria of transparency, it deserves to be uniquely identified. Some explanations are scrutable but not fully transparent. For example, in the case when a movie is recommended based on the number of times that a word appears in its description, there is no implication that the underlying algorithm is based on a Bayesian classifier. In such a case, we can imagine that a user attempts to scrutinize a recommender system, and manages to change their recommendations by modifying their ratings, but still does not understand exactly what happens within the system.

**Trust**

Previous studies indicate that transparency and the possibility of interaction with recommender systems increases user trust\[^6\]. A user may also be more confident in recommendations if he understands why a bad recommendation has been made. Trust could also be dependent on the accuracy of the recommendation algorithm. Transparency and trust are related, users feel more confident about recommendations that they perceive as transparent.

Trust also is seen as a long term relationship between a user and the organization that the recommender system represents. Some studies show that trust is associated with customer’s intention to transact, purchase a product and return to the website\[^7\]. Other abilities like security, privacy or reputation also build trust, and sometimes they are more important than the system’s ability to
Explain its result.

In general, competence perception about the system is an essential contribution to trust building. It provides trust-induced benefits such as the intention to purchase a recommended item, or to return and share with friends if it is a "good" website. Trust formation is also useful in making users feel that they exert less effort for the decision tasks and can reduce the uncertainty about the quality of a recommendation.

![Figure 3.4: Trust benefits](image)

**Effectiveness**

An effective explanation would help the user evaluate the quality of suggested items according to their own preferences. The goal of the recommender system is to help users make better decisions discarding irrelevant options. Effectiveness is the criterion that is most closely related to accuracy measures such as precision and recall.

**Persuasiveness**

Try to answer - Are the recommended items relevant for me? Although we can make precise recommendations for the user, if he does not perceive them to be important, the recommendation process is meaningless.

Sometimes a recommender may intentionally dwell on a product’s positive aspects and keep quiet about various negative aspects. But, it is also important to consider that too much persuasion may backfire once users realize that they have tried or bought items that they do not really want.

**Efficiency**

It is another usability principle which is based on reducing the time needed to take the decision if an item is the best for them or not. This criterion is one
of the most commonly addressed in the recommender systems literature given that the task of recommender systems is to find needles in haystacks of information. They are often used in conversational recommenders or knowledge based, where users continually interact with a recommender refining their preferences, and the explanations can be seen to be implicit in the dialogue.

**Satisfaction**

The presence of longer descriptions of individual items has been found to be positively correlated with both the perceived usefulness and the ease of use of the recommender system. When measuring satisfaction, one can directly ask users whether the system is enjoyable to use or if they like the explanations themselves.

### 3.2.2 Other goals

Together with the previously described goals, explanations can contribute to achieve other secondary or underlying goals. These can be:

**Intention to return**

If users possess a high perception of the recommender agent's competence, they would be more inclined to return to the agent for other product information and recommendations. However they would not necessarily intend to buy the product from the website where the recommendation was found. They are just looking for a trust worthy review.

**Education**

Educate users to help them better understand the product domain and its variety. Deep knowledge about the domain helps customers rethink their preferences and evaluate the pros and cons of different solutions. Eventually as customers become more informed, they are able to make wiser purchasing decisions. It may even awaken new desires they do not know yet.
3.3 Explanation models

There are many models of explanations depending on how the recommendation is explained and the amount of information used to explain it. A number of researchers [8] also reported results from evaluating explanation interfaces with real users, they demonstrated that a histogram with grouping of neighbour ratings was the most compelling explanation component among the studied users. Users perceive it as more capable and efficient in helping them interpret and process decision information and are more likely to return to it.

Recently new other ways to provide argumentations were introduced. The most relevant ones will be explained in this section.

3.3.1 Demographic

The background of this style is demographic information about the user and their ratings of items. The process is to try to identify users that are demographically similar to the target user and extrapolate from their ratings of items.

![Image](image_url)

Figure 3.5: A demographic explanation from the system INTRIGUE

3.3.2 Collaborative filtering explanations

These are one of the most used explanations. They are built on the assumption that the user’s peers with similar preferences can be exploited to derive items that might be of high interest. Although such a statistical learning mechanism is very popular for achieving accurate results, it is poor in explaining why a specific item is proposed. In the cases of Genius from apple, they explain its recommendations saying "Genius Mixes feature searches your iTunes library to find songs that go great together". Even though, they have been used in systems such as Amazon.com or MoviLens with some success.

Herlocker in [9] enumerated all the different kinds of explanations for collaborative Filtering. Here are some examples of commercial websites using this technique.
CHAPTER 3. EXPLANATIONS SYSTEMS

Figure 3.6: "Nearest neighbour" style, from Amazon.com

Figure 3.7: Genius by Apple - uses SVD algorithms and it is explained as "...songs that go great together"

Figure 3.8: "Neighbour Histogram" style - Used by MoviLens it was founded as one of the most effective style
3.3.3 Content based explanations, the "Why" style

The simplest strategy is to display the recommendation content in a rank ordered list with a "why" component for each recommendation explaining the results of the recommender system. This strategy is capable of explaining the propositions by their similarity to the set of preferred items in the user’s history and it has been used in several commercial websites together with the top-k recommender systems. It is used when the recommender systems require an input from the user.

Another approach could use the features of the previous rated item that appears in the recommendation. The template of this justification could be "This story received a high relevance score, because it contains the words f1, f2 and f3". This is a template of the so called "keyword" style.

Nevertheless these styles can not justify adequately their recommendations because they are based solely either on data about ratings or on content data which are extracted in the form of features that are derived from the items. To solve this, other approaches \(^1\) try to make a Hybrid system between Collaborative and Content based explanations. This explanations adapt what is known about the current user and the recommended item. An example of this style could be: "This item is suggested because it contains features $F_1$ and $F_2$ that are also included in items $X$, $Y$ and $Z$ that you like"

Other explanations called Tagsplanation uses tags as features to provide the explanation \(^2\). For instance, "We recommend the movie Kill Bill because it is tagged with quirky and you have enjoyed other movies tagged with quirky. Recommender systems typically do not consider the item fashion when generating explanations, Tagsplanations style is taken into account.

\(^1\)like CinemaScreen or Libra

\(^2\)
Figure 3.9: "keyword" style - Where the system associates the movies *The Dark Knight* and *Transformers* that influenced most the recommendation of movie *The Bourne Trilogy*.

Figure 3.10: "Why" Interface gives the user the reason why this product has been selected for him.

Figure 3.11: "Tagsplanation" for the movie *Eraserhead* - A list of all the Tags is provided.
3.3.4 Knowledge-based explanations, the Trade-off style

Also called constraint, organization interface or trade-off style. This technique exploits domain expertise like for instance If the user has need A then propose only items that possess property B to infer items whose characteristics match the preferences and needs of a user. However, only few recommender systems exploit explicit knowledge to derive item propositions and most use statistical learning techniques or even hybridize several different algorithmic approaches as CF or CB.

In Constraint-based systems only those constraints which are needed to generate the value assignments for entailing a recommendation are considered for explanations. The results are grouped according to their trade-off properties where the best matching item is displayed at the top of the interface along with several categories of trade-off alternatives. Each category explains how the items inside differ from the top candidate.

Sometimes the users are unlikely to have stated all of their preferences and they have not considered trade-off alternatives for the considered product. For this reason this technique can reduce the decision time and encourage users to consider new preferences.

Beside the wide spread approaches of collaborative and content-based filtering, knowledge-based recommender technologies gain an increasing importance due to their capability of deriving recommendations for complex products such as financial services, technical equipment or consumer goods. This seems to be significantly more effective in building user trust than the traditional algorithm. In the explanatory feedback conference [8] they proposed educating users about products by explaining which products exist instead of justifying why the system failed to produce a satisfactory outcome. With this technique the user is provided with partially satisfying solutions for his needs. Other study [10] shows that users could more quickly find their target choice when the recommended items inside each category were sorted by exchange rate rather than by similarity.
CHAPTER 3. EXPLANATIONS SYSTEMS

**Figure 3.12:** "Trade-off" Interface of a prototype of a laptop selling website

<table>
<thead>
<tr>
<th>Product</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ThinkPad T51 Notebook</strong></td>
<td></td>
</tr>
<tr>
<td><strong>VAIO VGN-NW780 Notebook</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Sony VAIO VGN-FZ29M/B Notebook</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.13:** Movie Tuner interface for Pulp Fiction - In this interface the user specifies if he wants less or more characteristics about the current movie

<table>
<thead>
<tr>
<th>Movie</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kill Bill, Vol. 1 (2001)</td>
<td></td>
</tr>
<tr>
<td>True Romance (1993)</td>
<td></td>
</tr>
<tr>
<td>Sin City (2005)</td>
<td></td>
</tr>
<tr>
<td>Killer, The (Die xue shuang xiong) (1989)</td>
<td></td>
</tr>
<tr>
<td>Grindhouse (2007)</td>
<td></td>
</tr>
</tbody>
</table>
In [10] it is described how to design a trade-off model and all the steps that must be followed. The main steps are:

1. **Generate all possible categories**
   Each trade-off property indicates whether the attribute of the recommendation is improved or compromised compared to the same attribute of the top candidate.

2. **Exclude dominated categories**
   To generate categories, never propose *dominating* categories. In other words if a category contains lighter and faster products, it will never be shown next to a category containing heavier and slower products.

3. **Select prominent categories with longer trade-off distance and higher diversity degree**
   The selected categories are diverse from each other and the contained products have overall stronger trade-off benefits.

4. **Rank recommendations**
   The recommendations are ranked by the exchange rate.

5. **Display**
   Show the top candidate and the items inside each category

**Used metrics**

To select which categories to show we need some metrics. The trade-off distance of each category is defined as the average sum of the exchange rate of all recommendations contained in the category:

\[
\text{Tradeoff Distance}(C_i, TC) = \frac{1}{|SR(C_i)|} |SR(C_i)| \sum_{R \in SR(C_i)} \text{ExRate}(R, TC)
\] (3.1)

Where \(TC\) is the top candidate, \(SR(C_i)\) is the set of recommendations contained in the category \(C_i\), and \(\text{ExRate}(R, TC)\) is the exchange rate of the recommendation \(R\) compared to the top candidate. A category possessing higher values in trade-off distance offers products with more gains than losses relative to the top candidate.

**Exchange rate. It is the potential gains versus losses compared to the top candidate.**

\[
\text{ExRate}(R, TC) = \sum_{i=1}^{p} w_i \cdot \text{exrate}(v_{r,i}, v_{tc,i})
\] (3.2)
The second category will be selected if it has the biggest value of \( F(C_i) \) which is the combination of the category’s trade-off distance and diversity degree with respect to the categories selected:

\[
F(C_i) = \text{Tradeoff Distance}(C_i, TC) \cdot \text{Diversity}(C_i, SC) \tag{3.3}
\]

where \( C_i \) is the current considered category in the remaining set, \( TC \) is the top candidate, and \( SC \) denotes the set of categories so far selected. The selection process end when the desired \( k \) categories have been selected.

- The global diversity of \( C_i \) with \( SC \) is the average sum of its local diversity with each category in the \( SC \) set. The local diversity of two categories is further determined by two factors: the title diversity and recommendation diversity.

\[
\text{Diversity}(C_i, SC) = \frac{1}{|SC|} \sum_{C_j \in SC} \text{TitleDiv}(C_i, C_j) \times \text{RecomDiv}(C_i, C_j) \tag{3.4}
\]

- The title diversity determines the degree of difference between the two category titles

\[
\text{TitleDiv}(C_i, C_j) = 1 - \frac{|C_i \cap C_j|}{|C_i|} \tag{3.5}
\]

- The recommendation diversity measures the different recommendations contained in the two compared categories. \( SR(C_i) \) represents the set of recommendations included in category \( C_i \).

\[
\text{RecomDiv}(C_i, C_j) = 1 - \frac{|SR(C_i) \cap SR(C_j)|}{|SR(C_i)|} \tag{3.6}
\]
3.3.5 **Hybrid Critiquing-based Recommender Systems**

This technique integrates the user self-motivated critiquing facility to compensate for the limitations of system-proposed critiques. The results show that the example critiquing system achieved better results in terms of user’s decision accuracy, cognitive effort and decision confidence.

It is based on a mix of trade-off model and content based. The interface allows the user to specify his preferences, but at the same time provides related products with the desired characteristics. It is not a complete system that explains the recommendations but it is more like a kind of a "user assistant" to find his necessities faster.

![Figure 3.14: Example of Hybrid interface](image)

The product found according to your preferences

**Canon PowerShot S2 IS Digital Camera**

*Price: $424.15*

- Canon: 5.3 M pixels, 12x optical zoom, 16 MB memory, 1.8 in screen size, 2.97 in thickness, 404.7 g weight

Adjust your preferences to find the right camera for you

- **Manufacturer**
  - Canon

- **Price**
  - $424.15

- **Resolution**
  - 5.3M pixels

- **Optical Zoom**
  - 12x

- **Removable Flash Memory**
  - 16 MB

- **LCD Screen Size**
  - 1.6 in

- **Thickness**
  - 2.07 in

- **Weight**
  - 404.7 g

We have more matching cameras with the following:

1. Less Optical Zoom and Thinner and Lighter Weight
2. Different Manufacturer and Lower Resolution and Cheaper
3. Larger Screen Size and More Memory and Heavier
3.4 Evaluating the impact of explanations

There are a few evaluations of the explanations in recommender systems and these evaluations were mainly focused on the users acceptance of the system. (See cap 15 of [4]). It is really difficult to evaluate it, because mainly we need feedback about the explanation from the user and sometimes either he does not want to provide it or is to lazy. Even though, we can distinguish between two ways of evaluation.

3.4.1 Subjective evaluation

Some experts [10] [11] during their surveys use a “form feedback” to evaluate different aspects such as:

**Overall system perceptions**

- *Recommendation quality*
  If the interface gives some really good recommendations.

- *Transparency*
  If the user can understand why the products were returned through the explanations in the interface. He should answer if he finds this interface useful to improve his shopping experience.

**Overall competence perceptions**

- *Perceived usefulness*
  If the interface is competent to help the user effectively and find products he really likes.

- *Perceived effort*
  If looking for a product using this interface required too much effort.

- *Satisfaction*
  If the overall satisfaction of the user with the interface is high.

**Trusting intentions**

- *Intention to purchase*
  If the user would purchase the product or just choose it if given the opportunity.

- *Intention to return*
  If the user in a future search would like to use an interface like this.

- *Decision Confidence*
  If the user is confident about the product he just selected and if it is really the best choice for him.
3.4.2 Mathematical evaluation

Additionally it was also necessary to find a way to evaluate the quality of explanations. The literature is not unanimous about this topic, for this reason adapting some exiting method was decided.

Precision and recall concern only the rating profile of a user “u” and measure the accuracy of “L”. However, precision and recall cannot distinguish between a relevant item from a more relevant item [12]. Other authors [13] [14] proposed that each item in a recommendation has to be evaluated according to a predefined set of interest dimensions. All of them agreed to use weights to measure the explaining performance of a item “i” in a determinate dimension. For instance, economy and quality will be used as simple example of interest dimensions to take a decision of the purchase of a laptop. A customer may have a higher interest in the dimension economy than in the dimension quality. For this reason the importance of weights dimensions have been directly specified by the customer.

To rank the explanations we can use the Explanation Utility ($e_a$):

\[
Explanation\ Utility(e_a) = \sum_{i=1}^{n} con(e_a, i) \cdot int(i)
\]  

(3.7)

Where $con(e_a, i)$ is the contribution of explanation $e_a$ to the interest dimension $i$ and $int$ specifies the degree of the interest of the customer in this dimension.

For each product that is a part of a recommendation, we have to present a set of explanations or argumentations which the customer finds interesting and fitting his needs. However, identifying the order of those arguments is not a easy task.

One approach is to position the most valuable explanations at the beginning and at the end of the ranking, because related studies indicated the existence of relations regarding the memorization of explanations when they are displayed in that way. The new formula is:

\[
Explanation\ Order\ Utility(e_{1..m}) = \sum_{pos=1}^{m} ExplanationUtility(e_{pos}) \cdot PositionUtility
\]  

(3.8)

<table>
<thead>
<tr>
<th>Explanation position</th>
<th>1º</th>
<th>2º</th>
<th>3º</th>
<th>4º</th>
<th>5º</th>
</tr>
</thead>
<tbody>
<tr>
<td>“position utility”</td>
<td>0.3</td>
<td>0.15</td>
<td>0.1</td>
<td>0.15</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 3.2: Utilities of explanation positions
With this we can order the explanations depending on the interest of the customer. For instance, if he is interested more in a popular and a well rated game than the price.

If we have \( n \) explanations, we have \( n! \) permutations of orders in which those explanations can be presented. The objective here is to identify the ordering which maximizes the Explanation Order Utility. For that reason identifying the order of those arguments is an important task.

We have not defined a way to distinguish between a relevant item and a more relevant item yet. To solve this problem, in [12] a user-oriented measure is proposed which is called explain coverage. They combine the justifications of all recommended movies in a single and ordered justification list:

\[
\text{Justification list}(J) = (f_1, c_{f_1}), \ldots, (f_m, c_{f_m})
\]

where \( f_m \) is the feature that can have each item and \( C_{f_m} \) denotes the overall frequency in the user list \( L \). For a user \( u \) that receives a recommendation list \( L \), the explain coverage for the justification list \( J \) is defined as:

\[
\text{Explain coverage} = \frac{\sum_{(f_i, c_{f_i})} \min\{C_{f_i}, P(u, f_i)\}}{\sum_{f_i \in F} P(u, f_i)}
\]  \hspace{1cm} (3.9)

It takes values in the range \([0,1]\) where the values close to 1 correspond to better coverage. This means that the justification is more effective, as the features that are included in the justification list \( J \) can be easily recognized and accepted by the user. Also it is a way to measure how much the features in the user feature profile are covered by the features of the items included in \( L \).
3.5 Design guidelines

The explanation process mainly comprises many steps:

- Determining which information should be included.
- How to organize the recommendation
- How to display it.

To deal with these steps, we should take into account some guidelines. After further research these are some of the most important guidelines extracted that should be followed in the design of an explanation engine.

1. Include actual products.
2. Recommending products in excluding groups enables users to reach their decisions much faster.
3. The modality and richness of an explanation interface did not seem to contribute much to the effectiveness of the interface.
4. Users prefer a short and concise conversational sentence for the low-risk products (movies, books, games...) and more detailed and informative explanation for high level of financial and emotional risks products (cars, houses, travels...)
5. Users appreciate long textual explanations only if the system conveys a strong confidence in its formulation.
6. In the trade-off explanation style, keep the number of trade-off attributes no more than three to avoid information overload and up to how many items to include in each category.
7. The model employed to give recommendations can also serve as a model for the argumentation.
8. The website's security, reputation, delivery service and privacy policy were also important considerations in building users trust.
3.6 Other aspects

In this chapter, the main aspects about explanations have been detailed. But there are also others that should be mentioned.

Presentation interface

The way explanations are shown can have an enormous impact on the final decision taken by the customer. Aspects like distribution, colors, fluency are factors that must be take into account when designing the user interface. Sometimes a decision is taken depending on the context in which it is represented.

Share personal information

If the user wants personal explanations he should first understand that he needs to share some personal information about tastes, ratings or purchases. Without this information, the system cannot predict accurate recommendations and the user should know it.

Potential risk

Modern recommenders are based on collaborative filtering: they use patterns learned from users’ behavior to make recommendations, usually in the form of related-items lists. The scale and complexity of these systems, along with the fact that their outputs reveal only relationships between items, may suggest that they pose no meaningful privacy risk. But this is not true, some authors have developed algorithms which take a moderate amount of auxiliary information about a customer and infer this customer transactions from temporal changes in the public outputs of a recommender system. With this attack they can predict the user’s purchases or his tastes.

Sales directives

Sometimes although the system could predict good recommendations for the user, these may be against marketing policy or other constraints. For example after the Apple and Samsung dispute, if you search on Amazon.com the phone "Samsung Galaxy SIII" and see the recommendations for this product then you will not see any comment about Apple phone. This does not occurs the other way around.
Chapter 4

Prototype design and implementation

In this chapter, the process of how the explanation engine are designed and carried out will be mapped out. The first steps are analysing the data to know which information is available, which is the behaviour of the users for recommended items and which factors contribute to the purchasing decision. Later a brief overview about the "Playstation Recommender engine" its predictions and its architecture will be given. Finally the implementation process of the explanation engine following the design principles explained in the state of the art will be detailed.

4.1 Data analysis

4.1.1 The Database

The data used for this project comes from the Playstation 2010 database. It contains logs about world users and information about the items available at this time. The main characteristics of the PlayStation Store® database are:

<table>
<thead>
<tr>
<th>PlayStation Store® Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users 17.343.542</td>
</tr>
<tr>
<td>Items 28.633</td>
</tr>
<tr>
<td>Ratings 331.163.176</td>
</tr>
</tbody>
</table>

Table 4.1: PSN main characteristics

This database is too large and work with the entire database could be computationally slow. I will test my explanation engine with different sets of data. For instance I will use items classified as comics purchased by users living in Europe or all the videos purchased in the USA.
CHAPTER 4. PROTOTYPE DESIGN AND IMPLEMENTATION

The metadata of the PlayStation Store® database is split in user’s metadata and item’s metadata. Within this data we can observe some demographic information.

1. User’s metadata: UserID, User, Gender, Age, Country, City

2. Item’s metadata: ItemID, Item, titleID, Genre, service-provider, free flag, title-name, Product-name, Short-description, Long-description

Figure 4.1: several figures about the database. We can observe how the users tend to give high ratings to the items. Rarely shown their dissatisfaction with some title.
4.1.2 User behaviour for recommended items

To know a little bit more about the user behaviour when they see recommended items, we have done a survey using the PlayStation Store® logs. Writing some Bash scripts, we have extracted the following information.

**Purchasing delay after see a recommendation:** the histogram show that there are two zones. In the first one, we can distinguish an important number of purchases within the first hour after see the recommendation. The second area is more dispersed and extends over time. For a better view, we split the analysis in five groups (see figure 4.3), where we can see that the 30.59% of the purchases was done in the first 45 minutes after receive the first recommendation. This indicates that, in general, users have a high impulse buying decision. Other conclusion could be the recommendation only reminds the item that the users already decided to buy. Regarding to the second area, it is too wide to extract single conclusions, there are many factors influencing the buying decision, perhaps the user needs to collect some money or ask for a second opinion about the item.

![Figure 4.2: Purchasing time after see the first recommendation. Note that the horizontal axis is in log(time [s]) since the range time is highly variable](image)

Figure 4.2: Purchasing time after see the first recommendation. Note that the horizontal axis is in log(time [s]) since the range time is highly variable.
Position of the recommended item in the list: Is the position of the item in the list important in the recommendation? The study confirms that items in the first 3 positions are the most purchased. These 3 positions represent the 54.7% of all purchases (see figure 4.4). This also confirms that the length of the recommended list is not relevant for the user as it seems that he only takes into account the top five items (see figure 4.5).
Number of recommendations shown before purchased: We have also analysed the number of recommendations shown to the user before the item is purchased. The measures show in average a purchased item needs to be recommended 4.95 times to the same user before the buying decision. This could indicate recommendations play a role as advertisement; if the user sees the same product many times this could generate a purchasing necessity.

4.1.3 Factors contributing to the purchase decision of games

Before implementing argumentations for the recommendations, we need to find out the main requirements about explanations in our game scenario. Information about this topic is reduced. Some specialized websites only shows a score based in own opinions and user votes. Another websites put a score using predefined parameters as graphics, sound, playability, duration or target. In [15] conclude the most important aspects influencing the decision for purchase a game seem to be:

<table>
<thead>
<tr>
<th>Low importance</th>
<th>Medium importance</th>
<th>High importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertising</td>
<td>Graphics</td>
<td>Price</td>
</tr>
<tr>
<td>Game content rating system</td>
<td>Developer</td>
<td>Friends recommendation</td>
</tr>
<tr>
<td>Publisher</td>
<td>Popularity of previous releases</td>
<td>Genre</td>
</tr>
<tr>
<td>Possibility to buy from the net</td>
<td>Material posted to internet</td>
<td>Theme</td>
</tr>
<tr>
<td>Post in forums</td>
<td>Demo played</td>
<td>Replayability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Experience with previous releases</td>
</tr>
</tbody>
</table>

Table 4.2: The most relevant factors in the game purchasing process

Following this pattern, arguments such as price, friend’s recommendation and genre are the most important. We must to take into account these factors when designing the explanation prototype.
4.2 The Recommender engine

The Recommendation Engine is the heart of the explanation system. It was designed by Sony speech and sound group and was improved during several internships along 3 years. It aims is to provide accurate and justifiable recommendations in the Play Station Network database. It can be configured in multiple combinations to generate recommendations using several algorithms, quality or satisfy time and memory constraints.

Figure 4.6: Current implemented algorithms
4.2.1 Input data

We dispose of the following input data:

**UI Matrix**: User-item relationships. This is the data about the liking, disliking or interests of users in items. This matrix has the following characteristics:

- Defined format (it can be either liked/disliked, a number of stars between 1 and 5, actions ID...) Sometimes, there are some implicit data available
- High sparsity
- It can be associated with a time signature
- The degree of "likeness" can be explicit (ratings) or implicit (purchased items, viewed demos, visited web pages...)

Generally speaking, we will have 2 types of UI Matrix:

- Ratings Matrix: In this case, we will have 3 types of relationship between user u and item i: positive (u likes i), negative (u dislikes i) and unknown (u does not know about i).
- Purchases Matrix: This matrix contains all the actions of our customers like "purchased" or "viewed". The problem here is that we do not dispose of negative relationships. Our matrix is filled with 1s (purchased) and 0s (not purchased). We might consider Purchased = positive relationship, but "Not Purchased" might be Dislike or Unknown relationship.
4.2.2 Architecture Description

These are the main steps of the recommender system.

Main Script

1. Initialization: add paths, charge parameters, start clock
   Now, for each split:

2. Load data (Ratings, Meta-data; binarize input data if necessary, initialize
   some global variables like total number of Items / Users)

3. Build sparse matrices

   Effect (the last one only in case of the Ratings Problem).
   global Effects ≡ function that computes the Global Effects sequentially
   and removes it from the Train Matrix.

   applyGE_mex: function that calculates in sequence the predictions for the
   systems GE0 (random), GE1 (popularity), GE2 (user effect) and evaluates
   them.

5. Build Similarity Matrices for CF2u, CF2i and CBFi
   Now, for each algorithm:

6. Compute predictions

7. Process predictions: add again the Global Effects, apply-if applicable-the
   novelty transformation (that is, penalize the predictions depending on the
   popularity of the movie).

8. Evaluate predictions resEval: This function can calculate the global accu-
   racy, top accuracy, top novelty and top variety. It might also compute the
   values of Precision, Recall and Fmeasure for both general and customer
   averaging.
   end (for each system)

9. Blending of predictions and evaluate them End (for each split)

10. Global Evaluation: give mean values of RMSE, RMSE@X, Novelty; plot
    the average PRC-RCL curves for all the splits
4.2.3 Predictions

After the computation, an output is given by the system. This output contains basically the vector with all the items which could be recommended for this user and other vectors with the score or prediction for the item vector depending of the selected algorithm. Note that this is only for one user.

\[
\begin{bmatrix}
458 & 0.21 & -0.8 & 1.5 \\
460 & 0.75 & -0.79 & 0.62 \\
464 & 0.01 & -0.01 & 1.01 \\
465 & 0.34 & 0.54 & 0.05 \\
\vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

Figure 4.7: Explanation Input Matrix

This matrix contains the results for the user i. The first column is the list of recommendable items, the next columns are the score for this items predicted by different algorithms (Cf2i, LSHCF2u and SVD).

For computational speed reasons, at the end of the recommender system we store all the available information about the active users and also the mapping vector of the items. This information is stored by the recommendation script and afterwards will be read by the explanation engine.

The recommender system is already designed and working properly for all the users who have an account in the Play Station Store. In the following page there are some screen-shots of a real recommendation.
Figure 4.8: Current recommendations for the PlayStation Store®. In the right side appear the list of recommended items, called "You may like"

Figure 4.9: If click on the list, the whole list appears
4.3 The Explanation System

The explanation system is an improvement, a complement to what we want to add to the recommendation system. We have designed it keeping these objectives:

- To establish trust with the user
- Provide useful explanations

The main idea is explain the recommendations using item lists and provide comprehensible explanations as “you will like this item because it has been well rated between your friends and people with similar tastes as you”. These are called "plain explanations". We will need to generate these lists which will contain all the most popular items in a determinate place, the best punctuated or these items which have been purchased for users with similar tastes. In a secondary implementation personalized information such as tastes, genre or friends purchases will be considered.

Moreover, another procedure to recommend items to the users using the “trade-off” technique has been developed. For comfort and to check the results, a graphical user interface (GUI) has been designed, using the tools provided by Matlab.

In this section we will give a general overview of how the explanations system runs. For a deeper vision and details about the implementation, see the Annex.

Figure 4.10: System sketch - In this sketch we can see how we are going to join the different module to get the final output: Recommendation + Explanation
4.3.1 Architecture Description

To design the explanation core I needed to modify the previous recommender system with some modifications to extract, manipulate and save all the necessary information. The main steps are:

Recommender script (main.m)

1. Save input data: In this part is saved all the data necessary to run the explanation core. It contains InfoUsersToEvaluate and explanationInput.

2. Generate the list: To compare if a given item belongs to a list, I need to calculate these list. The function generateExplanationsFileList([parameters]) generates the 14 different lists to explain the recommendations and store them.

Explanation script (ExplanationCore.m)

1. Load Input data

2. Initialization (read InfoUsersToEvaluate.mat and explanationInput.mat)

3. Load Basic lists

   Loop for all active users
   (a) Read item list for current user
   (b) Load specific lists (demographic and system lists)
   (c) Match explanations and calculate its "Explanation score Matrix"
   (d) Build the Explanation Matrix

   End for loop

4. Assign predefined templates for explanations to build text

5. Save the results in OutputCell and ExplanationVectorCell

6. Get item info to show in the GUI (Graphical User Interface)

7. Trade off set up

8. Call GUI
4.3.2 Lists generation
(generateExplanationsFileList.m)

With this code we generate the entire item list that later will be read by the main program. The lists are tables with 2 columns, the first is the Item ID and the second is the called “Explanation Score”. This value is a normalized number between 0.01 and 1 which give us an idea about the “strength” of the system regarding this explanation. The value 0.01 is for distinguish if there is an explanation for this item or not, in this case a 0 value is assigned. For instance if we obtain for a determined item that its “Explanation Score” is 0.876 in the popularity city list, means that likely this item is really popular in this place and therefore the explanation is “strong”. On the contrary if the score is 0.019 in the similar taste list means this explanation is faint for this user.

The system was designed to work with 3 different kind of list: basic, demographic and system list. In a secondary implementation we will add a “user’s knowledge” list.

**Basic lists**

Depending of the nature of the list, they are obtained directly from the primary predictions that the recommender system output. For example to generate the novelty list the result of the system is processed to obtain those items with the higher punctuation. The basic lists are formed by the following lists:

- **Surprising** It is a random Item which may surprise you. (Only if there is only one explanation for this item)
- **Essentials** The item is part of the “Essentials” collection
- **Price/Discount** The item has a special price
- **Trending/Boom** The item is a trending game!!
- **Novelty** The item is a new release
• **Popularity** The item is one of the “Top Download” games

• **Rating** The item is one of the best ever rated games

**Demographic list**
For these lists, we have to do a previous filter of the “uirTrain” matrix and extract which items are the most purchased depending on the variables as city, country, age or gender. The demographic lists are formed by the following lists:

• **City** The item is very popular in your city

• **Age** The item has been very well rated among people of your age

• **Gender** list this game is very popular among girls/boys

• **Country** The item is very popular in your country

**System lists**
We obtain these lists directly from the prediction of the recommendation system. Only the items with higher score in each case are selected and processed. The system lists are formed by the following lists:

• **CF2i** The item is similar to items that you have rated

• **LSHCF2u** The item was purchased by users with similar tastes to yours

• **SVD** Your interests suggest that you will like it
4.3.3 Trade-off Interface

To implement this technique we have selected four attributes following the guidelines of the section 4.1.3. These attributes are: popularity, novelty, rating and price. In this manner, if we compare the “Top Candidate” to all the items which are recommendable to the user, we can obtain 16 different kinds of categories in function of if the attribute depending on whether the attribute in question exceeds or not the attribute of the “Top Candidate”. Later we only need to calculate the Exchange rate as the mean of the difference vector.

<table>
<thead>
<tr>
<th></th>
<th>Novelty</th>
<th>Price</th>
<th>Rating</th>
<th>Popularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Candidate (TC)</td>
<td>0.3</td>
<td>0.7</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>item i</td>
<td>0.9</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Difference (TC-item i)</td>
<td>-0.6</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Binary difference</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: The item i belongs to the category 8 (0111 in binary). In this category are stored all the items which are better in price, rating and popularity.

Exchange rate = mean(TopCandidate – CurrentCandidate) = -0.05

I have slightly modified the algorithm described in the State of the art to adapt it to our system. The final implementation is:

**Trade-off algorithm**

1. Generate all the categories
2. Place the items in its category
3. Eliminate dominant categories
4. Calculate the TradeOffDistance of the categories
5. Rank the recommendations in favor of higher exchange rate
6. Show the categories with the items with the higher exchange rate

Due to technical reasons, all the information about the items and its attributes are not available at the moment of the implementation. For this reason we decided to implement a function called “preTradeoff” to simulate the distance matrix used in the algorithm. In future implementations this point should be done with real data.

In the next page there is pair of pictures about the GUI of the Explanation display showing all the parameters and the trade-off display.
CHAPTER 4. PROTOTYPE DESIGN AND IMPLEMENTATION

Figure 4.12: Explanation Display

Figure 4.13: Trade-off Display
4.4 Concept design

To have a general idea about the final result and to show the progress in different meetings we also have designed 2 screenshots of a concept design of the real system. These pictures are only a prototype.

Figure 4.14: Explanation Display - Concept design

Figure 4.15: Trade-off Display - Concept design
Chapter 5

Results

5.1 Objective evaluation measures

After design the system and check the code was working properly. We tested the explanation engine using different sizes of database. The first observed problem was how to check if the results obtained were good enough. Since with explanations we want to influence in the user behaviour and modify their purchasing conduct. This could not be quantifying easily. Therefore we have analysed the results using an objective point of view. We have defined some metrics as coverage or Explanation Score to compare the databases. On the other hand, also we have done a questionnaire between the members of the department to have a real opinion about the tastes of the costumers and observe if explanations can influence in the user behaviour.

The results have been the following: We have tested our scripts in the available database. As expected if we use large database the computational time grows up and it is hard to handle due to the excess of information. We have chosen 4 metrics:

- **Coverage.** Is the % of items that have at least 1 explanation among all the recommendable items for the user.

- **Coverage@20.** Is the % of items that have at least 1 explanation among the list of the 20 most likely items to be purchased for the user.

- **Number of explanation @10.** Is the average number of explanations that we obtain for the first 10 items of the recommendation list.

- **Explanation Score @10.** Is the average Explanation Score for the first 10 items of the recommendation list.

In all the cases we extract the average value between all the processed users. Note that these measures were meaningless to apply them in the trade-off format due to this style only provides alternative items.
CHAPTER 5. RESULTS

<table>
<thead>
<tr>
<th>Database</th>
<th>Coverage</th>
<th>Coverage @20</th>
<th>Number of explanation @10</th>
<th>Explanation Score @10</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSN10 JAPAN COMIC (300K ratings)</td>
<td>28.13%</td>
<td>89.73%</td>
<td>2.8</td>
<td>1.876</td>
</tr>
<tr>
<td>PSN10 EUROPE COMIC (800K ratings)</td>
<td>32.06%</td>
<td>93.28%</td>
<td>3.6</td>
<td>1.995</td>
</tr>
<tr>
<td>PSN10 USA VIDEO (9M9 ratings)</td>
<td>35.08%</td>
<td>94.47%</td>
<td>3.8</td>
<td>2.305</td>
</tr>
</tbody>
</table>

Table 5.1

In addition we were also interested to know the explanation distribution. We expect that the system would offer some variety in explanations and none of them were strengthened or weakened depending on their nature. We carry out histogram of explanations distribution based on their type. In the picture we can see how the explanations have uniform distribution and in the case of city popularity is somewhat lower because the number of cities in the database is very high and the lists cannot really specify whether an item is popular in the city.

![% Explanation distribution](image)

Figure 5.1: Explanation distribution histogram
5.2 Subjective evaluation measures

We have done a survey using an on-line questionnaire provided by google. In this questionnaire, 16 participants were asked about four explanation styles:

1. Single strong explanation
2. Multiple explanation
3. Explanation in paragraph
4. Trade-off explanation

To avoid influencing the participants about only video games, and other typical problems as participant doesn't like the explanation or monotony in the product; we decided to introduce new categories in the explanations, one asking about the same video-game, one asking about the same travel plan and the last one asking about random products. In each category we ask for the four styles of explanation. With this we will have a general overview about the participants tastes in several situations. Furthermore, after participants fill the first page of the questionnaire, they were asked to share their opinion about the explanations style and/or if they had some suggestion.

The results are varied. Trade-off style is the best accepted and the most liked. But followed by “several explanations”. In contrast the worst result is for the explanation in a paragraph. The users do not feel comfortable reading a lot of text. Regarding to the other styles is difficult to extract a general overview with this number of participants but seems some of them like strong explanation and others prefer several explanations.

Some participants complained about the phrase structure. They do not like phrases as “I would like to suggest / I would recommend you”. This is an important point because the use of repetitive phrase structure could irritate the users and lack their trust.
CHAPTER 5. RESULTS

Figure 5.2: Left - Results asked directly to the participants.
Chapter 6

Conclusions and future work

6.1 Conclusions

In this diploma thesis project, we present the design of an "explanation engine for recommender system" whose main objective is to explain the PlayStation Store® recommendations. The second objective is to establish trust and convince the users with its explanations, a very important factor in e-commerce environments.

Under these characteristics, we described a general overview about recommender systems (chapter 2) and we studied and evaluated the different kind of explanation styles proposed so far (chapter 3). From these studies, we extracted some design principles and guidelines to apply in our prototype. In parallel, we also investigated the user behaviour within the recommendation scenario analysing Logs of the PlayStation Store® database (section 4.1); and we selected the most decisive attributes in the purchasing decision: novelty, price, rating and popularity.

After these studies, we designed the prototype of an explanation engine. The innovative part of this engine is the fact that uses several information about the user to show him personal explanations tailored in his previous tastes (chapter 4.3). Besides, to visualize and to check if the engine is working properly, we created a graphical user interface (GUI) using Matlab tools. Also, to complete the system, we implemented a second method called “Trade-off” to provide alternatives to the selected item. We chose this style since it seems to be the one who strongly increase the user’s trust in the system.

The initial test showed the engine works properly. Even so, to evaluate the system, several objective metrics (coverage and explanation score) were studied and adapted to the system. Thus in the best scenario, the engine is able to explain up to the 35.08% of all recommendations for this user and up to 94.47% of the 20 most likely item list. Furthermore to check if the “trade-off” system it
was efficient, we did a questionnaire between some voluntaries and we checked this style it is one of the favourites. The questionnaire also shows explanations are well accepted in the purchasing process.

Finally we want to conclude, to establish user confidence not only depends on explanations itself, it must be an equal interaction between explanations, presentations and site prestige. However to do positive experience, the users must be motivated to contribute to other users with their own personal interest data. But this interaction also deals with fine-grained control over users personal data and profiles. To avoid this inconvenient user shall have full control of their public profile and sensitive privacy data.

6.2 Future work

This work presents the first step of an engine to explain recommendations in a real gaming scenario. Despite the substantial progress made during this thesis, there are a number of areas that require future research. These areas include:

1. Test the explanation engine in larger databases
2. Achieve more variety in the types of explanations
3. Use new methods for evaluating user satisfaction
4. Generate hybrid explanations blending existing explanations
5. Implement more complex explanation taking into account more sorts of input data


Appendix A

Tables
recommendations have been retrieved or computed. In other words, the explanation style for a given explanation may, or may not, reflect the underlying algorithm by which the recommendations are computed. There often is a divergence between how the recommendations are retrieved and the style of the given explanations. Consequently, this type of explanation would not be consistent with the goal of transparency, but may support other explanatory goals.

**Table 15.6:** Examples of explanations in commercial and academic systems, ordered by explanation style (case, collaborative, content, conversational, demographic and knowledge/utility-based).

<table>
<thead>
<tr>
<th>System</th>
<th>Example explanation</th>
<th>Explanation style</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSuggest-Usability [30]</td>
<td>See e.g. Figure 15.8</td>
<td>Case-based</td>
</tr>
<tr>
<td>LoveFilm.com</td>
<td>“Because you have selected or highly rated: Movie A”</td>
<td>Case-based</td>
</tr>
<tr>
<td>LibraryThing.com</td>
<td>“Recommended By User X for Book A”</td>
<td>Case-based</td>
</tr>
<tr>
<td>Netflix.com</td>
<td>A list of similar movies the user has rated highly in the past</td>
<td>Case-based</td>
</tr>
<tr>
<td>Amazon.com</td>
<td>“Customers Who Bought This Item Also Bought …”</td>
<td>Collaborative</td>
</tr>
<tr>
<td>LIBRA [11]</td>
<td>Keyword style (Tables 15.3 and 15.4); Neighbor style (Figure 15.3); Influence style (Figure 15.5)</td>
<td>Collaborative</td>
</tr>
<tr>
<td>MovieLens [29]</td>
<td>Histogram of neighbors (Figure 15.2) and Confidence display (Figure 15.5)</td>
<td>Collaborative</td>
</tr>
<tr>
<td>Amazon.com</td>
<td>“Recommended because you said you owned Book A”</td>
<td>Content-based</td>
</tr>
<tr>
<td>CHIP [20]</td>
<td>“Why is The Tailor’s Workshop recommended to you? Because it has the following themes in common with artworks that you like: *Everyday Life *Clothes…”</td>
<td>Content-based</td>
</tr>
<tr>
<td>Movieexplain [58]</td>
<td>See Table 15.7</td>
<td>Content-based</td>
</tr>
<tr>
<td>MovieLens: “Tags-planations” [62]</td>
<td>Tags ordered by relevance or preference (see Figure 15.7)</td>
<td>Content-based</td>
</tr>
<tr>
<td>News Dude [12]</td>
<td>“This story received a [high/low] relevance score, because it contains the words f1, f2, and f3.”</td>
<td>Content-based</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>System</th>
<th>Example explanation</th>
<th>Explanation style</th>
</tr>
</thead>
<tbody>
<tr>
<td>OkCupid.com</td>
<td>Graphs comparing two users according to dimensions such as “more introverted”; comparison of how users have answered different questions</td>
<td>Content-based</td>
</tr>
<tr>
<td>Pandora.com</td>
<td>“Based on what you’ve told us so far, we’re playing this track because it features a leisurely tempo . . .”</td>
<td>Content-based</td>
</tr>
<tr>
<td>Adaptive place Advisor [59]</td>
<td>Dialog e.g. “Where would you like to eat?” “Oh, maybe a cheap Indian place.”</td>
<td>Conversational</td>
</tr>
<tr>
<td>ACORN [65]</td>
<td>Dialog e.g. “What kind of movie do you feel like?” “I feel like watching a thriller.”</td>
<td>Conversational</td>
</tr>
<tr>
<td>INTRIGUE [6]</td>
<td>“For children it is much eye-catching, it requires low background knowledge, it requires a few seriousness and the visit is quite short. For yourself it is much eye-catching and it has high historical value. For impaired it is much eye-catching and it has high historical value.”</td>
<td>Demographic</td>
</tr>
<tr>
<td>Qwikshop [39]</td>
<td>“Less Memory and Lower Resolution and Cheaper”</td>
<td>Knowledge/utility-based</td>
</tr>
<tr>
<td>SASY [21]</td>
<td>“…because your profile has: *You are single; *You have a high budget” (Figure 15.1)</td>
<td>Knowledge/utility-based</td>
</tr>
<tr>
<td>Top Case [43]</td>
<td>“Case 574 differs from your query only in price and is the best case no matter what transport, duration, or accommodation you prefer”</td>
<td>Knowledge/utility-based</td>
</tr>
<tr>
<td>(Internet Provider) [23]</td>
<td>“This solution has been selected for the following reasons: *Webspace is available for this type of connection . . .” (Figure 15.4)</td>
<td>Knowledge/utility-based</td>
</tr>
<tr>
<td>”Organizational Structure” [49]</td>
<td>Structured overview: “We also recommend the following products because: *they are cheaper and lighter, but have lower processor speed.” (Figure ??)</td>
<td>Knowledge/utility-based</td>
</tr>
</tbody>
</table>

Continued on next page
Table 15.6 – continued from previous page

<table>
<thead>
<tr>
<th>System</th>
<th>Example explanation</th>
<th>Explanation style</th>
</tr>
</thead>
<tbody>
<tr>
<td>myCameraAdvisor</td>
<td>e.g “... cameras capable of taking pictures from very far away will be more expensive ...”</td>
<td>Knowledge/utility-based</td>
</tr>
<tr>
<td>[63]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transparency is not the only explanatory goal to consider when deciding upon explanation style. For example, for a given system one might find that users are more satisfied with content-based style explanations even though critique-based style explanations are more efficient. As of yet, there is little comparison between explanation styles with regard to their performance on explanatory goals. Only Hingston [30] has compared the understandability and scrutability of different explanation styles inspired by algorithm, although in these cases, the explanations were directly influenced by different underlying algorithms as well. Other studies have however considered the effects of different explanation interfaces on different explanatory goals [20, 29, 61].

Notwithstanding, the underlying algorithm of a recommender engine will to a certain degree influence the types of explanations that can be generated. Table 15.6 summarizes the most commonly used explanation styles (case-based, content-based, collaborative-based, demographic-based, knowledge and utility-based) with examples of each. In this section we describe each style: their assumed inputs, processes and generated explanations. For commercial systems where this information is not public, we offer educated guesses. While conversational systems are included in the Table, we consider conversational systems as more of an interaction style than a specific algorithm.

In the following sections we will give further examples of how explanation styles can be inspired by common algorithms as classified by Burke [13]. For each example we also mention how the recommendations are presented, and the interaction model that was chosen.

For describing the interface between the recommender system and explanation component we use the notation used in [13]: $\mathbf{U}$ is the set of users whose preferences are known, and $\mathbf{u} \in \mathbf{U}$ is the user for whom recommendations need to be generated. $\mathbf{I}$ is the set of items that can be recommended, and $\mathbf{i} \in \mathbf{I}$ is an item for which we would like to predict $\mathbf{u}$’s preferences.

### 15.7.1 Collaborative-Based Style Explanations

For collaborative-based style explanations the assumed input to the recommender engine are user $\mathbf{u}$’s ratings of items in $\mathbf{I}$. These ratings are used to identify users that are similar in ratings to $\mathbf{u}$. These similar users are often called “neighbors” as nearest-neighbors approaches are commonly used to compute similarity. Then, a
Appendix B

Questionnaire

B.1 Poll

This is the questionnaire made to evaluate the opinion of the voluntaries in front of recommendation plus explanations in different scenario.
Explanation Questionnaire

This is a small survey about explanations in recommender systems. The aim is to obtain some experimental results to improve the explanation engine for the PlayStation Store that I have been implementing during my master thesis.

Here you can see an example of recommendation in the PSN.
http://cdn.pocket-lint.com/images/dynamic/24011856a9b67ed57c8b32b306e91145eeef8e0a.jpg

Please focus in the explanation style and not in its content.
*Required

Personal information

Name/nick name *

Do you have any previous knowledge in recommender systems? *

○ yes
○ no

Questionnaire

A) Imagine the system recommends you the travel: Wine Country Touring And it gives you the following explanation: *
I would like to recommend you this travel because this is one of the newest places in the Hunter Valley. It has two highly regarded restaurants plus the boutique Tower Winery. And the rooms fit with your budget; they start at 350 coins per person per night.

1 2 3 4 5

I don't like it ○ ○ ○ ○ ○ I like it

B) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it proposes you: *
These are the best alternatives for this game: SONIC PINBALL 3D, RAYMAN ORIGINS, SIMS 3. But also I want to recommend you: SONIC 3 or Little Big Planet because these are popular and well rated but are an old release. - MEGAMAN10 or THE LEGEND OF SPYRO because these are popular and cheap but not highly rated

1 2 3 4 5

I don't like it ○ ○ ○ ○ ○ I like it

C) Imagine the LEVIS system recommends you: 505™ Straight Fit Trousers. And it gives you the following explanation: *
With thoughtful and timeless tailoring, our 505™ are the best option for you.

1 2 3 4 5

I don't like it ○ ○ ○ ○ ○ I like it

D) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation: *
I would like to recommend you this item because this item fits really well with your tastes, now is really popular and has a special price.
E) Imagine the system recommends you the travel: Wine Country Touring. And it proposes you the following alternatives:
- Prague: the historical pearl of Europe because it is popular, highly appreciated but out of your budget.
- Sun and beach in Mallorca because it is popular and well rated but is a typical destination.

F) Imagine the IKEA system recommends you: The table LACK. And it gives you the following explanation:
- Easy to assemble.
- Low weight.
- Easy to move.

G) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation:
- I would like to suggest you this item for the following reasons:
  - It is part of the “Essentials” collection.
  - It was purchased by users with similar tastes to you.
  - Your interest suggest that you will like it.
  - It has been very well rated among people of your age.
  - It is the most popular in your country.

H) Imagine the SWATCH system recommends you the CLOCK: Chrono Plastic Clock. And it gives you the following explanation:
The new series of vibrant wrist watches boasts a gorgeous and must-have selection of soft tones and hot colors, all designed to delight in different ways. Each of these ten novelties perfectly combines the sleek mixture of solid plastic and silicone.

I) Imagine the system recommends you the laptop: Sony Vaio Z series. And it proposes you the following alternatives:
- Lenovo ThinkPad because has cheaper price and more installed memory, although has slightly smaller hard drive capacity.
- MacBook Pro because has higher processor speed and different operating system, although they have slightly more expensive price.

J) Imagine the system recommends you the travel: Wine Country Touring. And it gives you the following explanation:
You never tried a trip of this style and your gastronomic taste point you might like this experience.

K) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation:
After analyzing your preferences the system is pretty sure that you will like it because this game is one of the best rated ever.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I don't like it</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. IMAGINE the system recommends you the travel: **Wine Country Touring**. And it gives you the following explanation: *

I would like to suggest you this travel for the following reasons: - You love wine - You are a single - You have a high budget

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>I don't like it</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Personal tastes**

In the previous section you have seen basically four styles of explanation:

A) Single strong explanation
B) List of several explanations (bullet points)
C) List of explanations in a paragraph
D) Explanation showing trade-off features with other games

**Which one do you prefer?** *

Put in order according to your preferences. (Please do not repeat position and do not go back in the form)

<table>
<thead>
<tr>
<th>1 - It is my favorite</th>
<th>2</th>
<th>3</th>
<th>4 - it is not my favorite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single strong explanation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Several explanations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explanation in paragraph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explanation showing trade-off features with other items</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you have any questions, ideas or suggestions, please do not hesitate to write it!

[Blank space for input]
B.2 Results

These are the results of the questionnaire. In the right side of the page are two columns. The first one is the number of participants who chose this option, the second column is the percentage overall.
K) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

J) Imagine the system recommends you the travel: Wine Country Touring. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>14%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
</tbody>
</table>

C) Imagine the LEVI's system recommends your 505™ Straight Fit Trousers. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
</tbody>
</table>

G) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>8</td>
<td>57%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
</tbody>
</table>

L) Imagine the system recommends you the travel: Wine Country Touring. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>29%</td>
</tr>
</tbody>
</table>

F) Imagine the IKEA system recommends you: The table LACK. And it gives you the following explanation:

- 1 - I don't like it
- 2
- 3
- 4
- 5 - I like it

<table>
<thead>
<tr>
<th>Rating</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>4</td>
<td>29%</td>
</tr>
</tbody>
</table>
D) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it gives you the following explanation:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>35%</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>50%</td>
</tr>
</tbody>
</table>

I don't like it  I like it

A) Imagine the system recommends you the travel: Wine Country Touring And it gives you the following explanation:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>36%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>29%</td>
</tr>
</tbody>
</table>

I don't like it  I like it

H) Imagine the system recommends you the CLOCK: Chrono Plastic Clock. And it gives you the following explanation:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>28%</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>7%</td>
</tr>
</tbody>
</table>

I don't like it  I like it

B) Imagine the system recommends you the game: SONIC The Hedgehog 4 Episode II. And it proposes you:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>14%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>57%</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>7%</td>
</tr>
</tbody>
</table>

I don't like it  I like it

E) Imagine the system recommends you the travel: Wine Country Touring. And it proposes you the following alternatives:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>21%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>29%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>29%</td>
</tr>
</tbody>
</table>

I don't like it  I like it

I) Imagine the system recommends you the laptop: Sony Vaio Z series And it proposes you the following alternatives:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>14%</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7%</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>36%</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>36%</td>
</tr>
</tbody>
</table>

I don't like it  I like it