Improving Performance for the Steam Recommender System using Achievement Data

Master Thesis – Data Science: Business and Governance

Tilburg University – School of Humanities

Author: Bram Notten

Student Number: u1239377

ANR: 451731

Supervisor: dr.ir. P.H.M. Spronck

Second Reader: dr.ing. S.C.J. Bakkes

Project period: January – August 2017

Project theme: Recommender Systems

Abstract

In modern society, most online shopping environments have a recommender system. Recommender systems use all data available to make a personalized prediction of a good or service someone might like. The current study will focus on improving an existing recommender system for the digital videogame marketplace Steam. Existing recommender systems use playtime and game ownership as a tool to recommend games to their users. In the current study, achievement information will be added to this list. Using Archetypal Analysis, similar users will be identified to predict a recommendation. Our research found that the addition of achievement data as we implemented, neither improved nor worsened the recommendations that the system gave in comparison with existing systems.

Table of Contents

1. Introduction

In the next chapter I will introduce the goal of this study. In section 1.1 I will give an overview of the context of this study. In section 1.2 I will state the research questions regarding the current study. Finally, in section 1.3 I will give an overview of the structure of this thesis.

1.1 Context

Customers are making an increasingly large amount of their purchases on the internet (Sleijpen & Wobma, 2013). In their meta-analysis, Chernev, Böckenholt and Goodman (2015) showed the existence of a phenomenon called choice overload. Choice overload is often explained as the effect that people buy less when there is more choice. Businesses want to avoid choice overload as it decreases the amount of purchases by customers. One can avoid choice overload by limiting customers' choices. This can be achieved with a recommender system. Recommender systems are algorithms designed to make a personalized recommendation for a good or a service. They are increasingly used in providing recommendations for digital goods, such as videogames, movies or music. Having a personal recommendation helps a customer making the right choice. This study will focus on recommendation of video games. It uses the popular video game platform Steam. Data will be gathered from Steam's digital marketplace.

Steam's digital marketplace sells videogames on Windows, Mac and Linux operating systems. It currently sells over 6000 different games (Hendrikman, 2015). With a market that has as many games as this, customers struggle to find a game that suits their needs. To have a game recommended based on your personal preferences could help in making this decision. A good recommender system therefore contributes in minimizing choice overload.

Sifa, Bauckhage & Drachen (2014) already designed a recommender system for Steam. The goal of the present study is improving their proposed recommender system by adding previously unexplored data. This will be done by building a new dataset using the Steam API. This new dataset will contain the same data as the dataset built by Sifa, Bauckhage & Drachen, supplemented with additional data on player achievements. This additional data will be used to attempt to improve the recommender system proposed by Sifa, Bauckhage & Drachen.

Sifa, Bauckhage & Drachen (2014) used Archetypal Analysis to design their recommender system. This analysis yielded better recall rates than several other algorithms (see chapters 2 and 3). In their study, they used playtime and game ownership to predict a list of games to be recommended to a user. In the current study, information on player achievements will be collected and added as a predictor.

Collecting new data and building a new dataset makes this study scientifically relevant. The data collected will be cleaned to suit the needs of this study. The raw, unclean data that was collected will be preserved as well. This gives other students and scientists the opportunity to use the new dataset for further research. Previous research on Steam's recommender system (Sifa, Bauckhage & Drachen, 2014) only included game ownership and playtime. Adding data on player achievements in addition to the data on game ownership and playtime might prove useful in increasing the quality of the Steam recommender system. In addition, the study by Sifa, Bauckhage and Drachen will be replicated, which possibly increases the relevance of their results.

Next to the aforementioned scientific relevance, upgrading the recommender system also has a practical application. Having a personalized recommendation of games one should buy or play does not only help the customer in making a better choice, it also helps game developers finding users that may be interested in their games. Making good use of this information could therefore boost the sales of their games. The fact that recommender systems in the gaming community are relatively new contributes to the study's practical relevance.

In the next section, the research questions will be discussed. These will be the main focus for the rest of the thesis.

1.2 Research Questions

With the goal of building a new dataset which includes data on game achievements the problem statement is as follows:

Can the Steam recommender system be improved by using additional features in the form of achievement information?

With this problem statement, the research questions will be as follows:

1. *What is a suitable data structure for analyzing recommender systems?*
2. *Can a recommender system based on Archetypal Analysis be replicated with new data?*

3. Can the recommender system based on Archetypal Analysis be improved using additional information on achievements?

1.3 Structure

In this section I will give an overview of the structure of the rest of the thesis. In chapter 2 related work regarding the subject will be discussed. This will be a set up to answer my research questions. In chapter 3 I will discuss the experimental setup. In this chapter everything regarding the collection of the data, the dataset itself and the analysis will be explained. In chapter 4, the results of the analysis will be stated. In chapter 5 the results that have been found, will be discussed. Finally, I will conclude this thesis by answering the research questions.

2. Related work

In the following chapter I will delve into related work on the subject matter of recommender systems. In section 2.1 the concept of recommender systems will be introduced. In section 2.2 I will give an overview of the game analytics field itself. In section 2.3 I will dive deeper into recommender systems for Steam exclusively. In section 2.4 I will explain what achievements are and connect them to the current study. Finally, in section 2.5 I will give an overview of the statistical methods used in this thesis.

2.1 Choice Overload and Recommender Systems

People often have too many choices which results in fewer purchases. This phenomenon is called choice overload. As mentioned in their meta-analysis, Chernev, Böckenholt and Goodman (2015) analyzed 99 experiments regarding choice overload. They showed that four factors facilitate choice overload. These factors are: higher choice set complexity, greater decision task difficulty, higher preference uncertainty and an unclear decision goal. They finally conclude that although many studies contradict the effects of choice overload, it is indeed an existing phenomenon.

A solution for the choice overload problem is a recommender system. A recommender system tries to predict and recommend a list of one or several options specific to what a user might like (Cremonesi, Koren & Turrin, 2010). A recommender system takes the four factors facilitating choice overload and tries to minimize them. Taking the TV-series and movies streaming service Netflix as example: Netflix uses a recommender system to make a prediction in the form of a list of series and movies which the user might like. All sorts of data are used to compile this list. The goal of a recommender system is thus is to predict a “match” between the options available and the user’s preference. When these two correspond, a recommendation can be made.

In 2006, Netflix held a contest for machine learning experts to improve the accuracy of their recommender system (Bennet & Lanning, 2007). They publicly released a dataset containing 100 million anonymous movie ratings. This dataset was meant for machine learning and computer science communities to work with. The goal of this competition was to improve their algorithm that predicts the recommendations of movies and TV-series. They awarded a big prize for the team that could improve the accuracy of their current algorithm by 10 percent. This example shows the practical relevance of machine learning algorithms but also the fact that machine learning and (to a certain extent) recommender systems are becoming an important topic for commercial companies, which they are willing to pay large amounts of money for.

Throughout the last two decades, recommender systems have been studied extensively. In their comparative analysis, Lu, et al. (2015) studied several kinds of recommender systems and analyzed their developments. One of the first recommender systems used a collaborative filtering method. Collaborative filtering assumes that when one person likes A and B and another person also likes A, then this person will have a higher chance liking B in comparison with another random person. Lu et al. concluded that collaborative filtering methods are still among the most popularly used recommender systems.

Another practical example is the study by Linden, Smith & York (2003). In their industry report, they examined the web shop Amazon.com, which was one of the first websites to integrate a recommender system. Their goal was to explain how its recommender system works. They used an item-to-item collaborative filtering method to match each of the user's purchased and rated items to similar items. Item-to-item collaborative filtering is a technique that matches the user's purchased items to similar items rather than matching a user to similar users (Deshpande & Karypsis, 2004). Linden, Smith & York then combined these items into a recommendation list. This example proved to be an effective form of marketing targeted at the customer. Linden, Smith & York predicted the further development of recommender systems in the future.

Another example to show the effect of item-based collaborative filtering was examined by Sarwar et al. (2001). They compared the effect of item-collaborative filtering methods in recommender systems with other techniques. They found that with datasets becoming larger, it became harder and more time consuming to make recommendations. They concluded that item-based recommender techniques were dramatically better then simpler techniques such as k-nearest neighbor.

In a study by Cremonesi, Koren & Turrin (2010), an attempt was made to measure the performance of recommender algorithms on top-N recommendation tasks (giving a user a top-N ranked list of possible best matches). They examined the differences between top-N and other used recommender systems. The difference between the recommender systems they examined was in the distinction between accuracy metrics (precision/recall) and error metrics (RMSE/MAE). They found that, while the error metrics were more popular and dominant in scientific literature, accuracy metrics may be a more natural benchmark because they directly measure the quality of top-N recommendations. Illustrating this, they found that a naive non-personalized algorithm could reach almost similar accuracy scores as sophisticated algorithms using error metric optimization.

Returning to the problem of choice overload, it is important to take set size and set quality into account. Bollen et al. (2010) found that when someone is recommended a larger set of good recommendations (20 recommendations to choose from) this does not necessarily lead to higher choice satisfaction. Due to the increased choice difficulty, it is better to keep the recommendation set smaller – limited to 5 recommendations – to get the best choice satisfaction. It is therefore important to keep the set of recommendations to choose from small, when designing a recommender system.

The current study will combine an item-to-item collaborative filtering algorithm with a regular collaborative filtering algorithm. Archetypal analysis will be item-to-item based and looking for similar users is based on regular collaborative filtering.

2.2 Game Analytics

As seen in the previous section, recommender systems have been studied extensively; however, the application of recommender systems for video games is relatively new. The field of game analytics is a specific domain in analytics that analyzes data gathered from video games. In the words of Nasr, Drachen & Canossa (2016), it focuses on the game as a product as well as the means of providing a good user experience. They suggest three forms of collected data. First of all, customer metrics contain all information about the user as a customer. A good example of a customer metric is the price for which a game has been purchased. The second type of metric they discuss are community metrics, which encompass the community as a whole. Community forum activity for instance, is considered a community metric. The last type of metric they propose is a gameplay metric. Gameplay metrics include all the information about the user as a player, such as player achievements, object interaction etc. In the field of game analytics, these three metrics are most often used.

An interesting branch of research in the field of game analytics commits itself to finding behavioral patterns of players in a videogame. These behavioral patterns can be used to make relevant predictions. For example: in their study, Bauckhage et al. (2012) could predict churn, i.e., when a player would stop playing a game, using data on how long the game is played initially. In a study by Drachen et al. (2012) players were clustered to create behavioral profiles of these players. Clustering players via different analyses proved to be very successful in discovering patterns in player behavior.

On a large scale, cross-game player behavior analysis on Steam, Sifa, Drachen & Bauckhage (2015) tried to find undiscovered patterns of gaming behavior focused on the time users spent playing individual games. They used a clustering method to group players into different player profiles with time spent playing as the most important factor. Their study showed that the majority of the users – about two thirds – played only one game, and the rest of the users equally divided their time playing over several games. In the discussion section of their article they mention the benefit of using player profiling in recommender systems. Player profiling in the form of archetypal analysis will be used in the current study to build the recommender system.

2.3 Steam Recommender System

With insights gained from the previous sections, the combination of recommender systems and the field of game analytics was explored by Sifa, Bauckhage & Drachen (2014). They were the first to create a recommender system using customer metrics gathered from Steam. Steam is a marketplace that sells digital copies of videogames on Windows, Linux and Mac operating systems. It currently sells over 6000 games (Hendrikman, 2015) and has over 125 million users. With a peak of 8.9 million users using the service at the same time, it is the biggest digital game distribution platform for buying games on Windows, Linux and Mac (Saed, 2015). It therefore makes sense to look at Steam for this specific application.

Sifa, Bauckhage & Drachen (2014) used Archetypal Analysis to design a recommender system for Steam. Archetypal Analysis is a clustering method that instead of finding clusters based on average values, finds clusters based on extreme values. (More on Archetypal Analysis will follow in the next section). In their study, a list of 101 games was made. One-hundred of these games were selected out of a pool of the 500 most played games on Steam. Next to this, one other game was selected. This other game was one of the top 5 games played by a particular user. The playtime of this game was set to zero minutes. This blinded game was the focus of the recommender algorithm – their goal was to let their recommender system find out that this was a game that should be recommended to the user. Subsequently a predicted playtime was calculated for the 101 games. They calculated this predicted playtime using data on playtime of the most similar users in comparison with the user in question. They then ranked these games on predicted playtime to check if the game that was blinded returned in the top-L (L varying between 5 and 20) of this ranking. L in top-L stands for the number of games in the list of best predictions. For example, with an L of 5, the top 5 best predicted games will be recommended to the user in question. If the game was in the top-L there was a hit (100% recall rate), otherwise there was a miss (0% recall rate).

Recall rate was used as a way to evaluate their results. A recall rate is an off-line evaluation method that is used because it simulates scenarios where games are offered in an unordered way. At the time of the writing of this paper, in the Steam recommender system results are presented this way. To verify their results, they compared their Archetypal recommender system to four other recommender systems: nearest neighbor, a model regarding general popularity of games, a model regarding total playtime and a random baseline. They showed that the Archetypal based model had the highest recall rates in comparison with the other four systems using Top-L models.

2.4 Steam Achievements

In their research, Sifa, Bauckhage & Drachen (2014) only used the consumer metrics game ownership and playtime to make a recommender system. Additional data could however improve their findings. Other data maintained by Steam are data on achievements. Hamari & Eranti (2011) defined player achievements as “an optional challenge provided by a meta-game that is independent of a single game session and yields possible reward”. Achievements thus are optional in the completion of a game. According to Hamari & Eranti (2011) different kinds of achievements exist. Some are obtained automatically when you enter a certain phase of the game while others require great amounts of “grinding” (completing repetitive and time-consuming tasks within the game). Player achievement info could therefore be useful for clustering players into different player profiles. When someone has completed more achievements than other players this does not automatically mean that this player spent more time in a game. It could however mean that this player belongs to another player archetype. In the current study, achievement information will be collected and added to the Archetypal recommender system used by Sifa, Bauckhage & Drachen (2014) to attempt to improve the recall rate and thus better recommendations.

2.5 Statistical Methods

In the following section statistical methods used in the current study will be briefly explained. In section 2.5.1 Archetypal Analysis will be discussed. In section 2.5.2 I will discuss cosine similarity.

2.5.1 Archetypal Analysis

As mentioned in the previous section, Archetypal Analysis is a clustering method that, instead of finding clusters based on average values, finds clusters based on extreme values. This is done by decomposing a given matrix into a collection of extreme entities (Cutler & Breiman, 1994). These extreme entities are called archetypes. Archetypal Analysis also calculates stochastic coefficient vectors which represent all data entries as a convex combination of archetypes, with each number representing a data entry’s score on one of the archetypes. A convex combination occurs when all the numbers in a vector sum up to one. So, for example: in the Archetypal game recommender system by Sifa, Bauckhage & Drachen (2014) they transform the huge user-playtime matrix so that every user represents one vector with only a number of values based on the number of archetypes that have been calculated. When comparing Archetypal Analysis to a clustering method based on means, it is harder to calculate good recommendations using the latter, as the extreme entities are flattened to a mean. Because these archetypes are clusters of extreme entities, Archetypal Analysis is very suitable to apply to recommender systems. For a more detailed information on Archetypal Analysis, I refer to Cutler & Breiman (1994) and Sifa, Bauckhage & Drachen (2014).

2.5.2 Cosine Similarity

The second important statistical method I use is cosine similarity. Cosine similarity is a statistical way to measure similarity (or distance) between two vectors. It uses the cosine of the angle between the two vectors to measure similarity (Huang, 2008). Cosine similarity between two vectors can be found by taking the dot product of the two vectors.

$$sim = \frac{v1v2}{\|v1\|\|v2\|}$$

The highest scores are the most similar vectors. In the current study, cosine similarity will be used to find the most similar users to the user we want to make a recommendation for. It is also used in calculating predicted playtimes for the user that gets a recommendation.

Having discussed related work and relevant statistical methods I will now turn to deeper discussion of the methodology of the study.

3. Method

In the following chapter I will discuss the experimental setup and the method. Section 3.1 will cover all aspects concerning the dataset. In section 3.2 the experimental procedure will be discussed. Section 3.3 will discuss the use of achievement data in the current study. Finally, section 3.4 will cover how the analysis will be evaluated.

3.1 Dataset

In the following few sections, all aspects concerning the dataset will be covered. In section 3.1.1 I will cover the collection of the data. In section 3.1.2 the cleaning of the dataset will be discussed. Finally in section 3.1.3 the content of the dataset will be discussed.

3.1.1 Collecting the Data

Collecting new data and building a new dataset is a vital part of this study. The original data set of Sifa, Bauckhage & Drachen (2014) could not be used, as it only contained playtime data. New data has been collected through the Steam API. The Steam web API serves as a tool for developers to use data maintained by Steam. To gain access to this API a special license must be applied for. When this application has been approved, an API-key is retrieved to access the Steam web API. Another element needed to access Steam data via the Steam web API is the Steam ID of the user you would like to request data of. The Steam user ID is an identification number, unique to every user on Steam. A list of 100,000 Steam IDs was externally supplied to build the database. With these two prerequisites, data could now be requested one at a time via the Steam web API. To make many requests in a short amount of time, a scrape program was written in the programming language Python. This program quickly requests data for the given Steam ID and then saves it to a dictionary. This scrape program reduced the time to request data for all 100,000 supplied Steam IDs.

The first dataset that was built contained data on playtime and ownership of games. When requesting this data, a list of owned games with their playtime in minutes was returned. This data was requested for the full list of 100,000 Steam user IDs. It took three to four hours to collect the first dataset using the Steam API. This led to a database of 38,000 user entries. The 62,000 users that were lost during this process were because data could not be requested for these users. One of the reasons for this is that many Steam users choose to keep their data private and can therefore not be requested by the Steam API.

This first database was now extended with data on achievements for each game per user. This was done in the same way as the first dataset with one difference. The distinction was that in addition to the Steam user ID, an application ID had to be supplied as well. The application ID is an identification number unique for each game available on Steam. The achievement data that was returned, contained a list of all achievements and their corresponding titles for each game. It also returned whether the given user had accomplished these achievements or not. The complete dataset now contained information on playtime, ownership and achievements (for every user, on every owned game).

3.1.2 Cleaning the Dataset

The dataset was now cleaned to meet the requirements of the current study. All players who owned fewer than 5 games were removed from the dataset. Also, all games that had a cumulative playtime of less than 25 hours over all users, were removed due to their unpopularity. The data on achievements was transformed into a percentage. This percentage represents achievements accomplished by a user to the total number of achievements for a game. The dataset finally had the following structure: for every user there was a list of owned games and for every game the dataset included the playtime for this game and the percentage achievements accomplished.

3.1.3 Content of the Dataset

The two datasets, one containing playtime information and one containing achievement information, had to be requested separately, but were combined into a single dataset. This big dataset had the following structure: it was one big Python dictionary with the keys being user IDs and the values being the games of this user. The games of a user were also structured in a Python dictionary. Here, the keys were the app IDs of the games owned by this user. The values for these games consisted of the playtime on this game, total achievements available for this game and achievements obtained. For the games that had information on achievements, it was possible to see for each unique achievement if it was obtained or not. It was therefore easy to derive how many achievements were obtained and how many there were in total per game. This was the structure of the database before analysis.

This final database contained 13,033 unique users and 4,673 unique games. 2,780 of these games had information on achievements. The database was saved as a numpy array, as this is an easy format to work with. It contained 1,096,145,046 minutes of playtime in total. On average, games were played 1,961.6 minutes. Only games that had a playtime higher than 0 minutes were included to get to this average. Playtime ranged from 1 minute to 1,131,822 minutes on a single game for a single user. On average, users obtained 12 achievements per game for games that hold achievements. The minimum number of achievements available was 1, the maximum number of achievements available for a single game was 1,708. Table 1 shows some descriptive statistics about the dataset. Figure 1 shows the relationship between all games of all users in the dataset (from least played to most played) and their playtime.

	Average per user	*Minimum*	*Maximum*	*Standard deviation*	*Median*
Playtime	1961.6*	1	1,131,822	9869.725	231
Achievements	12	1	1,708	87.289	30

Table 1: Database Descriptives, playtime in minutes.*

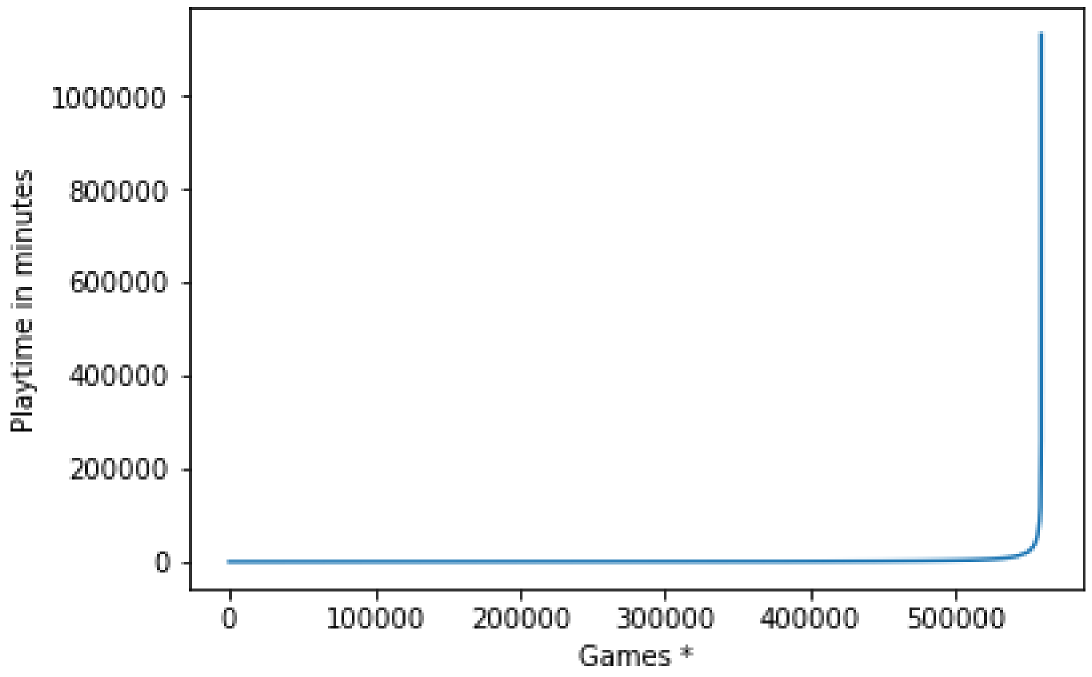

Figure 1: Playtime in minutes per game. *Games sorted from lowest playtime to largest

3.2 Experimental Procedure.

In the current study, two experiments, both with two analyses were conducted. The first experiment used Archetypal Analysis to calculate 9 archetypes per user. A second experiment calculated 50 archetypes per user. This second experiment was conducted to see if a higher amount of archetypes leads to a higher recall rate as already shown by Sifa, Bauckhage & Drachen (2014). The two experiments both contained the following two analyses. The first analysis is a replication of Sifa, Bauckhage & Drachen's (2014) study. The second analysis was essentially the same as the first apart from the fact that additional data on achievements were added to calculate the archetypes. The results of these two analyses will finally be compared to see if there is a difference. The procedure for the two analyses was the same.

The procedure of the two analyses was as follows: first, Archetypal Analysis transformed the two large matrices (one with and one without achievement information) into stochastic coefficient vectors with 9 and 50 scores per user for the two experiments respectively. This was done for both the playtime and achievement matrix. These two k-values were chosen because they best represent the values used by Sifa, Bauckhage & Drachen (2014) without having to do too many repetitions. Next, a list of 100 randomly chosen Steam IDs was made to test the algorithm. For every user in this list, the 5 most similar users were then calculated using cosine similarity. Additionally, for every user in the list, the top 3 most played games were selected and one was randomly chosen to be blinded. This blinded game should be recommended back to this user again. The blinding was accomplished by changing the playtime of this game to 0. Next, 100 of the top-500 most played games in the dataset were randomly selected and placed in a list with the blinded game. Consequently, a predicted playtime was calculated for all 101 games. With all 101 games having a predicted playtime, a ranking was made. When the blinded game was returned in the top-L of this ranking, there was a hit. The aforementioned procedure is one trial in the progress. Finally, all the hits divided by the trials gave the recall rate for one repetition. Every analysis contained 400 repetitions of 100 trials. The next subsections will zoom in further on the procedure, step by step. The complete Python code can be found in Appendix A.

3.2.1 Archetypal Analysis

To work with the data, the dataset had to be transformed into a large 13,033 by 4,673 matrix. In this matrix, each row represents a user and each column represents a game. For the first analysis, only the playtime data was used and therefore every entry is a user's playtime (in minutes) for a specific game. The first step was finding a top-5 of most similar users. This was done using cosine similarity. One of the reasons Archetypal Analysis was used is because the calculation of 13,033 cosine similarities for every user is a time-consuming process. Archetypal Analysis dramatically reduced the number of columns for every user. Following Sifa, Bauckhage & Drachen (2014), I chose to do an Archetypal Analysis consisting of 9 and 50 archetypes. After conducting Archetypal Analysis, a vector of 9 and 50 scores was returned for every user. These vectors represent convex combinations of scores on each of the 9 and 50 archetypes. Summed up, every vector will be equal to one. Thanks to these shorter vectors (9 and 50 columns instead of 4673) it is much faster to find the 5 most similar users.

For the calculation of the archetypes a Python module named PyMF (Python Matrix Factorization) was used (Thureau, 2010). PyMF is a module that contains several types of matrix factorization utilities, including Archetypal Analysis. The module is compatible with Python version 2.6 and it is one of the fastest ways to perform Archetypal Analysis in Python.

The calculation of the archetypes was the most time-consuming process of this study. For the first experiment, the first analysis took 6-7 hours to complete the calculation of the archetypes. The second analysis took a lot longer since it was almost twice as big. This took about 24 hours total. For the second experiment, the calculation of the first analysis took about 24 hours. The second analysis took about 144 hours. The calculation of archetypes thus takes significantly longer when more archetypes are calculated and when the dataset is larger. The analyses were done on a desktop PC containing a 4.0 gigahertz quad core processor and 16 gigabytes of RAM memory.

3.2.2 Most Similar Users

With all the users now represented as a convex combination of the 9 archetypes, finding the most similar ones was a fast process. Because of the relatively small number of users in the dataset (13,033) it was decided to find the top-5 similar users from the whole dataset instead of using a smaller sample. This was done to obtain more accurate results. After finding the top-5 most similar users the recommendation process could be entered.

3.2.3 Making a Recommendation

To see if a game is to be recommended to a user, a ranking of 101 games was made. This ranking was based on predicted playtime for these 101 games. The 101 games consist of 100 games randomly selected from the top-500 most played games in the dataset. The other game is one of the top-3 most played games of the user for which we are going to make a recommendation. In the current study we take this game, set its playtime to zero minutes and predict a “new” playtime for this game, along with the 100 other randomly selected games. When we have a predicted playtime for this user on the 101 games, the games will be ordered on playtime. This ordered list, or ranking, was used to recommend the top-L games for the user in question. If the blinded game returns in the top-L recommendation there is a hit.

3.2.4 Predicting Playtime

To make a prediction of playtime for the 101 games in the list, the following formula (as used by Sifa, Bauckhage & Drachen, (2014)) was used:

$$\hat{t}_{ji} = \frac{\sum_{\mathrm{u} \in \mathrm{U}} Sim(P_i, P_u) t_{ju}}{\sum_{\mathrm{u} \in \mathrm{U}} Sim(P_i, P_u)}$$

In this function, $\hat{t}_{ji}$ is the predicted playtime for user j, game i. Next to this, u stands for one of the top-5 most similar players to player j. The sum of the cosine similarities between player i and player u times the playtime of player u on game j, divided by the sum of all the cosine similarities player i and player u, gives the predicted playtime for game i by player j.

3.2.5 Evaluation and Recall Rate

To evaluate the analysis, it is checked whether the blinded game is returned in the top-L recommendation list. If a game is returned, there is a recall rate of 100%. If the game is not returned in the top-L list the recall rate is 0%. Taking the mean of all these recall rates will give the total recall rate. Recall rates are used as an evaluation method here because it simulates the process of a user receiving an unordered list of game recommendations.

3.3 Achievements

Specific to the current study is the addition of achievement data. The only difference with the playtime data analysis was that for the database with achievement data, new archetypes would be calculated. The achievement data was supplied in the form of a percentage. This percentage stands for the number of achievements completed in relation to all achievements available for one specific game. Before calculating the new archetypes, a correlation between time played and achievement percentage was calculated. If this correlation was too high it was not worth calculating new archetypes because the outcome would most likely be the same. If there was a low correlation, new archetypes would be calculated which would then be used to do the same calculation for finding recommendations as mentioned in the previous section.

For the calculation of the new archetypes a matrix of 13,033 by 9,346 was supplied. This matrix included all the playtime data and achievement percentage data per user in the dataset. For the calculation of the archetypes, again a k of 9 and 50 were used so the calculation returned a matrix of 13033 by 9 and 13033 by 50 – which again is significantly smaller than the dataset before Archetypal Analysis.

3.4 Final Evaluation

To evaluate if there was a difference between the two methods used in the previous sections, the two recall rates were compared to see if they differed significantly. To get more accurate results, 400 repetitions of the analysis were run. One analysis made recommendations for 100 users for both methods (playtime vs. playtime with achievement percentages). To see if they differ significantly, the mean, the standard deviation and the standard error of the mean were calculated for the 400 repetitions. When the two recall rates did not fall within the same range of the standard error of the mean, there was a significant difference between the two methods.

4. Results

In the following section, results found on the analyses are discussed. Section 4.1 will discuss the correlation between playtime and achievement data, Section 4.2 will discuss the recall rates found for both recommender systems.

4.1 Correlation between Playtime Data and Achievement Data

A Pearson correlation was conducted to see if playtime and achievement data have a high linear coherence. There was a small positive correlation between the two variables, $r = 0.266$, $p = 0.000$. Finding a correlation is not unexpected, as longer playtimes tend to indicate more achievements collected. However, the correlation was low enough to expect that achievement information may add extra useful information, and therefore there was sufficient reason to continue the analysis.

4.2 Recall Rates

To compare the two recommendation methods, the mean of the recall rates, the standard deviation and the standard error of the mean were calculated. This was done over 400 repetitions for both experiments on both analyses. First the results for the experiment with $k = 9$ will be discussed. The playtime analysis resulted in a mean recall of 82.158 percent. The playtime analysis had a standard deviation of 4.454 and a mean standard error of 0.223. The achievement analysis resulted in a mean recall of 82.095 percent, had a standard deviation of 4.090 and had a standard error of the mean of 0.205. The results are visualized in figure 1. Sifa, Bauckhage and Drachen (2015) found a recall rate with a mean of 90% with similar parameters. The only difference between the current study and the study by Sifa, Bauckhage & Drachen (2014), was that their database is significantly bigger.

Because the two recall rates did not differ significantly with a k of 9, another analysis with a k of 50 was conducted. This k of 50 was also used in the analysis by Sifa, Bauckhage & Drachen (2014) and resulted in higher recall rates compared with a k of 9. In the current study, a k of 50 yielded the following recall rates as a result. For the playtime data only a recall rate of 85.423 percent was found, it had a standard deviation of 4.073 and a mean standard error off 0.204. The analysis on achievement data resulted in a recall rate of 85.283 percent with a standard deviation of 3.383 and a standard error of the mean of 0.169. These two recall rates again did not differ significantly. All the results are visualized in Figure 2. The comparison of results between the current study and Sifa, Bauckhage & Drachen’s study are shown in table 2.

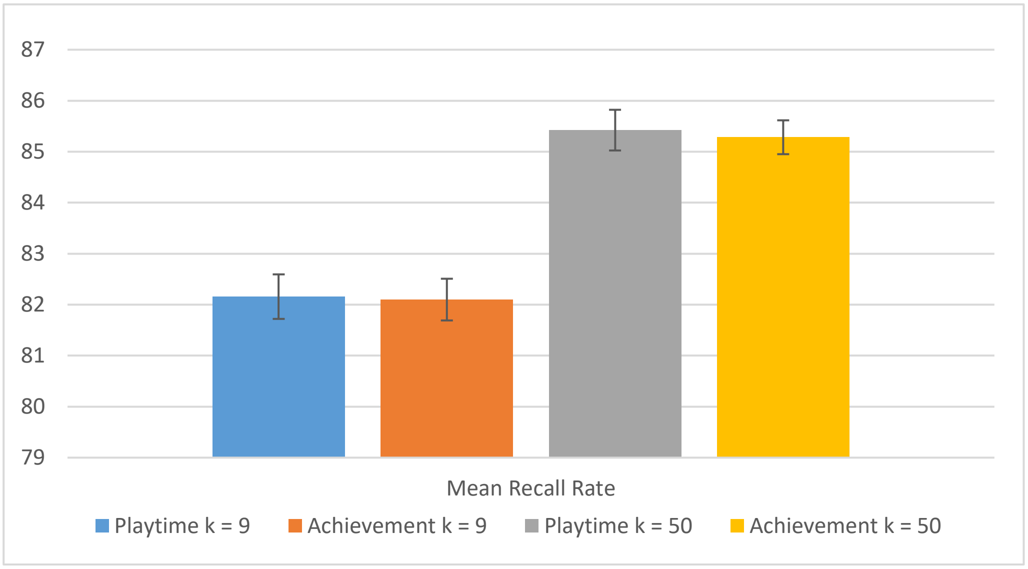

Figure 1: Mean recall and standard error of the mean.

	K = 9	*K = 50*
Sifa, Bauckhage & Drachen	around 90 %	around 94 %
Current Study Playtime	82.157 %	85.423 %
Current Study Playtime + Achievements	82.099 %	85.283 %

Table 2: Comparison of results between 2 studies

5. Discussion

In this chapter, I will evaluate the goal of the current study. In section 5.1 I will discuss my findings and in section 5.2 I will discuss the size of the dataset. In section 5.3 I will look deeper into the data collected on achievements and how it can be used to extract more information. Finally, in section 5.4 I will look back at the scientific and practical relevance of this study.

The current study had two main goals. The first goal of this study was to build a new dataset of Steam users and their owned games, with information on playtime and achievements. The second goal of this study was to try and improve an already existing recommender system with the addition of extra data. The idea behind this was that with more data, the algorithm would be better in finding differences between users and generally have more accurate results. To achieve this goal, a suitable data structure had to be found. In their research, Sifa, Bauckhage & Drachen (2014), used Archetypal Analysis to make their recommender system and they succeeded in finding desirable results using Archetypal Analysis to construct their data structure. In this study, Archetypal Analysis was also used as a tool to find similarity between users. Replicating their study and comparing these results to a second analysis (which had the same structure, only with extra achievement data) was the key method that was used to achieve the second goal.

5.1 Findings and Results

In order to assess the validity of the proposed analysis in this study, a Pearson r correlation test between playtime and achievement data was carried out. This was done to make sure that the achievement data that was added to improve the recommender system did not have a high correlation with playtime data. If this correlation would have been too high, there was no reason for continuing with the main analysis because the achievement data would not give much more information over playtime data. In the current study there was a small positive correlation between playtime and achievement. This was small enough to continue with the main analysis. For the main analysis, two recall rates were calculated. One recall rate for the playtime dataset and one recall rate for the achievement dataset. These recall rates were calculated over 400 repetitions of the algorithm making 100 recommendations. The recall rates tell how many of the blinded games are returned in the top-20 games as calculated by the algorithm. The recall rates did not differ between the analyses for both experiments. There was a difference in the recall rates between the different experiments. For the second experiment, with a k of 50 there were significantly higher recall rates than the recall rates in experiment one with a k of 9. Because the two recall rates for achievement and playtime did not differ in both analyses does not mean that the research questions are to be rejected all together. There were some limitations that could have influenced the insignificant difference between the results of the two methods, as will be discussed below.

5.2 Size of the Dataset

The final dataset for the current study contained over 13,000 users and over 4,000 games. This is relatively small in comparison with the dataset used by Sifa, Bauckhage & Drachen (2014). Their dataset contained over 6 million users. With an analysis looking for similar users like the one used in the current study, more data means more accurate results. In particular, we can assume that for users who have an eccentric taste in games, Sifa, Bauckhage & Drachen's data set will have been able to find better matches using their archetypes. In the current study, lower recall rates were found in comparison with the analysis on the dataset used by Sifa, Bauckhage & Drachen. This is most likely due to the fact that the two datasets differ in size.

5.3 Extracting More Information from Achievement Data

Another limitation in this study was the way the achievement data was used in the analysis. The simplest form of achievement data was used in this study, namely, percentage achievements achieved. This reduction was made to keep the dataset manageable. However, the information gathered on achievements can be interpreted to express more information than only the percentage of achievements achieved. As this is the first time achievement data was used for an analysis like this, the choice was made to use it in its simplest form. Other ways to use the achievement data to give the data more meaning will be discussed below.

A way to extract more information from the achievement data could be a weighted achievement system. What I mean by this is that rarer or harder achievements weigh heavier in the calculation of the archetypes. A way to do this is to calculate, for every achievement, the percentage of people that accomplished it and let this percentage then be a score on the particular achievement. Summing up all the achievement scores per game per user then gives a weighted user achievement score. The higher this score, the rarer the achievements will be that a user has accomplished in a game. This could distinguish between users who only get the achievements that are scored by everyone, and the users who are so invested in a game that they also get the rare achievements. Using some kind of weighting in the achievement data makes the data more meaningful and could lead to better results.

5.4 Scientific and Practical Relevance

The current study is, although no results were found that improved upon earlier research, scientifically relevant. By building a new database and examining an expanded form of the already existing Steam recommender system, new insights were gained. First of all, we found that achievement percentage information does not correlate very strongly with playtime. This means that it will always be interesting to study their effects on recommender systems in other ways than the current study has done. Also, by using a dataset with the size of the one used in the current study, no significant differences were found between the two methods of recommendation. A larger dataset could help in finding better results. As for the practical relevance, it has already been stated by Sifa, Bauckhage & Drachen (2015), with thousands of games releasing every year, it would be practical for the player, as well as the developer to have a good, working recommender system.

6. Conclusion

In this final chapter, I will answer my research questions to formulate a conclusion. I will also give directions for future research.

1. What is a suitable data structure for analyzing recommender systems?
In their study, Sifa, Bauckhage & Drachen (2015) showed that a recommender system based on Archetypal Analysis was a viable way analyzing recommender systems. Taking recall as an evaluation metric proved to be useful as well. According to Cremonesi, Koren and Turrin (2010) accuracy metric gave significantly better predictions in comparison with error metrics such as RSME.

2. Can a recommender system based on Archetypal Analysis be replicated with new data?
In principle, a recommender system based on Archetypal Analysis, can be replicated using new data. Although the results on the current study are somewhat lower (87% recall on the study by Sifa, Bauckhage & Drachen (2014) and around 82% on the current study with a k of 9), the numbers found were indicative for previous results found; the difference is likely the result of using a much smaller dataset. The fact that the recall rates were higher with a larger value of k is also in line with the findings of Sifa, Bauckhage & Drachen.

3. Can the recommender system based on Archetypal Analysis be improved using additional information on achievements?
In the current study, I did not succeed in improving the playtime based recommender system with the additional achievement information, expressed as a percentage of achievements scored per game. With the recall means respectively being 82.158 % (playtime) and 82.095% (achievement) for experiment 1 and 85.423% (playtime) and 85.283% (achievement) for experiment 2 no significant difference between the two analyses in the two experiments was found. It should be noted that in this study achievement data were used in a rather simple form, and that achievement data may still be shown to improve recommendations when used in a more meaningful form.

Problem statement: *Can the Steam recommender system be improved by using additional features in the form of achievement information?*
With the results in the current study I cannot fully answer this question. The results showed no sign of improvement over the analysis containing only playtime. However, with a correlation of 0.266 and the knowledge that achievements encompass relevant information on players, it is not shown that there is no

potential for achievement data to improve recommendations. Future research should delve deeper into the matter of Steam achievements in relation to the recommendation of games on Steam

References

Bauckhage, C., Kersting, K., Sifa, R., Thurau, C., Drachen, A., & Canossa, A. (2012, September). How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *Computational Intelligence and Games (CIG), 2012 IEEE conference on* (pp. 139-146). IEEE.

Bennett, J., & Lanning, S. (2007). *The Netflix prize. In Proceedings of KDD cup and workshop* (Vol. 2007, p. 35).

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., & Graus, M. (2010, September). *Understanding choice overload in recommender systems. In Proceedings of the fourth ACM conference on Recommender systems* (pp. 63-70). ACM.

Chernev, A., Böckenholt, U., & Goodman, J. (2015). *Choice overload: a conceptual review and meta-analysis. Journal of Consumer Psychology, 25*(2), 333–358

Cremonesi, P., Koren, Y., & Turrin, R. (2010, September). *Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of the fourth ACM conference on Recommender systems* (pp. 39-46). ACM.

Cutler, A., & Breiman, L. (1994). *Archetypal Analysis. Technometrics, 36*(4), 338-347

De Myttenaere, A., Golden, B., Grand, B. L., & Rossi, F. (2015). Study of a bias in the offline evaluation of a recommendation algorithm. *arXiv preprint arXiv:1511.01280*.

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS), 22*(1), 143-177.

Drachen, A., Sifa, R., Bauckhage, C., & Thurau, C. (2012, September). Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on* (pp. 163-170). IEEE.

Hamari, J., & Eranti, V. (2011). *Framework for designing and evaluating game achievements. Proc. DiGRA 2011: Think Design Play, 115*(115), 122-134.

Hendrikman, M. (2015, August 28). Steam heeft meer dan 6000 games. Retrieved from: https://tweakers.net/nieuws/105000/steam-heeft-meer-dan-6000-games.html

Huang, A. (2008, April). Similarity measures for text document clustering. *In Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand* (pp. 49-56).

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

Lu, Jie, et al. *"Recommender system application developments: a survey." Decision Support Systems 74* (2015): 12-32.

El-Nasr, M. S., Drachen, A., & Canossa, A. (2016). *Game analytics*. Springer London Limited.

Saed, S. (2015). Steam has over 125 million active users, 8.9M concurrent peak. Retrieved from: https://www.vg247.com/2015/02/24/steam-has-over-125-million-active-users-8-9mconcurrent-peak/

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295). ACM.

Sifa, R., Bauckhage, C., & Drachen, A. (2014). *Archetypal Game Recommender Systems*. *In LWA* (pp. 45-56).

Sifa, R., Drachen, A., & Bauckhage, C. (2015). *Large-scale cross-game player behavior analysis on steam. Borderlands, 2*, 46-378

Sleijpen, G., & Wobma, E. (2013, July 1st). Steeds meer mensen kopen online. Retrieved from: https://www.cbs.nl/nl-nl/nieuws/2013/27/steeds-meer-mensen-kopen-online

Thureau, C. (2010). Python Matrix Factorization [Computer Software].

Appendix A

```
### Totale Analyse

import numpy as np
import pandas as pd
import sklearn
from sklearn import metrics
import random
from scipy import spatial
import sys

### Inladen Archetypes + 100 vergelijkbare spelers

archetypes_pt = np.load("archetypes_pt_data.npy")
archetypes_ach = np.load("archetypes_ach_data.npy")
data = np.load("data_matrix_pt_26-6.npy")

top500 = np.load("top500_list.npy")
top500 = top500.tolist()

### Kiezen Speler voor aanbeveling en kies 10 uit 100 vergelijkingen

def top_10_compared_users(user_number, archetypes):
    compared_user_list = []
    #for i in comparison_users_1000:
    for i in range(0,13033):
        y = sklearn.metrics.pairwise.cosine_similarity(archetypes[i].reshape(1, -1),
archetypes[user_number].reshape(1, -1))
        compared_user_list.append((y[0][0], i))
    compared_user_list.sort(reverse = True)
    if compared_user_list[0][0] == 1:
        del compared_user_list[0]
    top10list = []
    for i in range(0, len(compared_user_list)):
        if len(top10list) == 5:
            break
        else:
            top10list.append(compared_user_list[i][1])
    return(top10list)




### Kiezen game voor reccommendation en op 0 zetten

def choose_rec_game(user_number):
    toplist = []
    for i in range(0, len(data[345])):
        toplist.append((data[345][i], i))
    toplist.sort(reverse=True)
    top5list = []
    for i in range(0,5):
        top5list.append(toplist[i][1])
    pred_game = random.choice(top5list)
    data[user_number][pred_game] = 0
    return(pred_game)


### Kiezen 100 games uit top500 voor ranking

def top101(top500, rec_game):
    top = []
    top.append(rec_game)
    while len(top) <51:
        x = random.choice(top500)
        if x in top:
            continue
        else:
            top.append(x)
    return(top)


### uitrekenen verwachte speeltijd per game
def top_L_ranking_pt(user_number, rec_game, archetypes, top, top10list):
    pred_playtimelist = []
    for i in top:
        simlist = []
        simlist_times_playtime = []
        for x in top10list:
            sim = sklearn.metrics.pairwise.cosine_similarity(archetypes[x].reshape(1, -1),
archetypes[user_number].reshape(1, -1))
            #print("^^", sim[0][0], data[x][i])
            pt_sim = sim[0][0] * data[x][i]
            simlist.append(sim[0][0])
            simlist_times_playtime.append(pt_sim)
        try:
            pred_pt = (float(sum(simlist_times_playtime)/sum(simlist)))
            #print("pred_pt:", pred_pt)
        except ZeroDivisionError:
            pred_pt = 0
        #except:
        #    print(sys.exc_info()[0])
        try:
            pred_playtimelist.append((pred_pt, i))
        except TypeError:
            print("Typerror", i)
            pred_playtimelist.append((pred_pt, i))
    pred_playtimelist.sort(reverse=True)
    for i in pred_playtimelist:
        if i[1] == rec_game:
            rec_game_playtime = i[0]

    return(pred_playtimelist.index((rec_game_playtime, rec_game)))

#user_number = 345
#data = np.load("data_matrix_pt_26-6.npy")
#top10list = top_10_compared_users(user_number)
#rec_game = choose_rec_game(user_number)
#top = top101(top500, rec_game)
#reccomendation = top_L_ranking_pt(user_number, rec_game)

def user_list(tot_users):
    user_number_list = []
    while len(user_number_list) < tot_users:
        number = random.randint(0,13032)
        if number in user_number_list:
            continue
        else:
            user_number_list.append(number)
    return(user_number_list)

user_number_list = user_list(100)

def make_rec(toplist_number, archetypes, x):

    rec_tuple = [0,0]
    for i in user_number_list:
        data = np.load("data_matrix_pt_26-6.npy")
        top10list = top_10_compared_users(i, archetypes)
        rec_game = choose_rec_game(i)
        top = top101(top500, rec_game)
        try:
            reccomendation = top_L_ranking_pt(i, rec_game, archetypes, top, top10list)
        except ValueError:
            continue
        if reccomendation < toplist_number:
            rec_tuple[1] += 1
            rec_tuple[0] += 1
        else:
            rec_tuple[0] += 1
        print(rec_tuple[0], x)
    return(rec_tuple)
```

```
user_number_list = user_list(100)

achlist = []
ptlist = []

for i in range(0,25):
    user_number_list = user_list(100)
    x = make_rec(20, archetypes_pt, i)
    y = make_rec(20, archetypes_ach, i)
    ptlist.append(x)
    achlist.append(y)
```