OULUN AMMATTIKORKEAKOULU

Kristian Ratia

ANALYZING MOVIE TRENDS IN GAMES BASED ON PUBLIC DATA

ANALYZING MOVIE TRENDS IN GAMES BASED ON PUBLIC DATA

Kristian Ratia
Bachelor’s Thesis
Spring 2019
Information Technology
Oulu University of Applied Sciences

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä: Kristian Ratia
Opinnäytetyön nimi: Elokuvien trendien analysointi peleissä, perustuen julkiseen dataan
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: Kevät 2019 Sivumäärä: 36

Pelin tekemiseen menee viikosta useampaan vuoteen. Ison ja viimeistellyn pelin tekeminen maksaa paljon. Isot peliyritykset tutkivat tarkasti markkinoita, jotta heidän pelinsä menestyisi mahdollisimman hyvin. Pienillä peliyrityksillä ei ole rahaa eikä työvoimaa tutkia markkinoita. Elokuva-ala tutkii markkinoita hyvin tarkasti, lisäksi elokuva-ala käyttää paljon rahaa markkinointiin ja aloittaa markkinoinnin hyvissä ajoin ennen elokuvan julkaisua.

Tässä opinnäytetyössä tutkitaan miten elokuvien trendit vaikuttavat pelien trendeihin. Tutkimuksen perusteella elokuvien trendit näkyvät pelien trendeissä. Tässä työssä tutkitaan vain elokuvien ja pelien määriä, ei sitä miten trendit näkyvät tuloksessa rahallisesti. Jotkut trendit korreloivat erittäin vahvasti, toiset taas heikosti. Tulevista elokuvista voidaan päätellä jossain määrin millaisia pelejä on tulossa.

Työtä varten ladattiin suuri määrä julkista dataa elokuvista ja peleistä. Datan käsittelyä varten asennettiin serveri joka ajoi MySQL tietokantaa. Työssä käytettiin Linux-serveriä, MySQL-tietokantaa ja useita Linuxin apuohjelmia tiedon käsittelyyn.

Asiasanat: IMDb, Steam, Game industry, Movie industry

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author: Kristian Ratia
Title of thesis: Analyzing movie trends in games based on public data
Supervisor: Kari Laitinen
Term and year when the thesis was submitted: Spring 2019 Pages: 36

Making a game will take time from a couple of months to a couple of years. The Time needed depends on how large and polished the game will be. Big game companies can use hundreds of millions dollars to make a game while small indie game companies use only a couple of thousands of dollars. There is a huge difference between game budgets, but every company has the same problem: predicting if the game will sell. Big companies will use their resources for the market research. Making the research will take time and cost money.

The movie industry is another big industry that uses a lot of money for making an entertainment product and also uses a lot of money to research which movies could succeed. Big movie studios begin marketing new movies as soon as possible. It is possible to know what kinds of movies are coming next year.

This thesis studies whether there are any correlation between movies and games. It is done by comparing movie tags to games tags. Tags are downloaded from IMDb and Steam which are public databases. This thesis study only does the amount of tags correlate together. It does not study whether the income of movies and games correlate together.

This thesis proves that there is a correlation between the amount of movie and game tags. An average correlation is strong enough so that predicting incoming games from incoming movies is possible.

Keywords: IMDb, Steam, Game industry, Movie industry

PREFACE

I would like to thank my family for support, Oulu Game Lab for knowledge, for a place where I wrote my thesis and for great connections.

Oulu, 23.10.2018

Kristian Ratia

TABLE OF CONTENTS

VOCABULARY

AAA	Games made by middle sized or big game companies are called AAA-games. AAA-games have typically higher development and marketing budget.
CSV	Comma-separated values are text files that have information separated with a comma. For example, Excel can easily make a sheet with this file.
HTML	Hypertext Markup Language
IMDb	Internet Movie Database is a net site that lists 5 million movies
IP address	Internet Protocol address is a numerical label that is given to every device which is connected to the internet.
Steam	Steam is a platform that sells games.
URL	A Uniform Resource Locator is a web address.
VPS	Virtual Private Server is a cheaper option for a dedicated physical server. On VPS companies share physical hardware for multiple VPS users. Every VPS user has own copy of the operating system.

1 INTRODUCTION

Small game studios do not have enough resources to examine what are upcoming trends in gaming. Big studios and movie studios can use a lot of money for researching what topics are hot in the future. Making an AAA-game or big Hollywood movie costs more than $100 million. The cost of making a high-quality product is so high that they want to decrease risk anyway possible. Doing good background studies is one of the ways how to reduce risk. Another way to reduce risks is by starting the marketing of a new film or game as soon as possible. In this thesis, those two are the main things. By checking what kinds of movies are coming, small studios can predict what upcoming trends are. If they do affect, then small studios do not have to make their own research. They can just check out what movies will come in the future and use that data when designing a game. This thesis does not study if movies affect the sales of games. This thesis only studies the number of games and movies. Studying does trends affect sales could be a topic for another thesis. Studying does trends correlate is large enough a topic for one thesis.

1.1 Internet movie database

IMDb is a website that lists a lot of movies and TV series. There are more than 5 million different titles on IMDb. IMDb lists several different things from titles e.g. Site list cast, production crew, plot summaries, fan reviews, ratings and plot keywords. IMDb has made a dataset which has basic information about movies and staff. IMDb has listed those in .tsv (tab-separated values) files. IMDb's dataset does not list plot keywords. Plot keywords can be downloaded from movies plot keyword pages. On the IMDb website, the user can register to the site and after that, they can add information to the site. There might be some fake information on the IMDb site but most of the information is trustful. Plot keywords are harder to upkeep because different people want to pinpoint different things from the plot. IMDb has thought that and on plot keywords, people can agree or disagree on the keyword. With those votes, tags that are fake can be separated from tags that are relevant. Most of the pages do not have any tag words. There are only 13,000 titles that have tag words.

1.2 Steam

Steam is a popular digital distribution platform for PC games. Steam has 125 million users. There were 20,489 games on Steam on 10.1.2018 [1]. Getting sales numbers is quite hard for Steam games. This thesis does not study sales, but sales numbers are needed when studying whether incoming movies affect games income. In this study, only Steam games, are studied, the sales figures of which can be found [2] in Orland's article. The company which owns Steam is Valve. Valve does not tell how many copies games have sold, but because of the achievement system, people could calculate precisely how much each game had sold. Steam changed their system how they inform achievements and people cannot use that anymore for calculating the sale amounts of the games. Games used in this thesis are those 13,000 games that are Kyle Orland listed in the article. Steam does not have a different page on game tags. Steam lists tags on the same page where it lists other information about the game. Game data used in this thesis is from Steam game pages which is public information.

1.3 Timeline

Steam was released in 2004. The Steam achievement system was released on 10.10.2007. Before that date, there are no accurate game sale figures. There are some sale figures before that because some game studios, which have published their game earlier, have made achievements to their games, but there are a lot of games that have been released before the achievement system and which have not been updated to use achievements. Therefore, this thesis only studies games published between 2008 and 2018.

1.4 Studies made before

Studies that would compare movies and games could not be found, but there are some studies that tried to predict which movie will be a hit in the future. Studies that try to predict the success of movies usually used several different variables for studying which movies make money. Studies used variables, such as the director, actors, genre, rating, plot synopsis, an average annual profit of movie industry and release date. Predicting success of movies is a very difficult thing to do. “None of the studies thus far have succeeded in suggesting a model good enough to be used in the industry.” [3]. Studies that tried to predict the success of movies used machine learning.

Studies made about predicting a movie revenue show that predicting a movie revenue is very difficult and movie studios make various bad investments. Only 36% of movies had a box office revenue higher than production costs [4]. In this thesis, it does not matter that most movies do not make a profit. Even if the movie fails, studios still use a great amount of money on marketing and the topic of the movie will be more popular. This study will show if the number of movie topics will affect the number of games on the same topic.

2 TOOLS

2.1 Server

A server was needed for this study. The server was rented from Contabo and it had 6 cores CPU, a 30-GB ram and a 600-GB SSD hard disk. The server had a Debian Linux operating system. Because of handling a large number of files, which took a couple of weeks, the server was needed. Another option would have been to use a home computer but in some cases, this would need the computer to be up and running for a couple of weeks and it is not comfortable to have a home computer that uses 100% of computing power all the time. With the server, there are not any noise or heating problems.

For this study, using Linux as a servers operating system, made things a lot easier. With Linux, it is easy to make scripts that will do most of the work. There are several small free programs for Linux that help to handle a large amount of data.

2.2 wget

Wget is a computer program that downloads content from web servers. Wget is a simple program which does only what it is programmed to do: to download content. There is a huge amount of data on IMDb and it would have taken over 60 days to get all the data needed with one instance of wget. One instance of wget uses only one core. On a processor which has six cores, one instance uses only 16.7% of the processor computing power. To the full computing power of processors, it needed to run six instances of wget at the same time.

Because over 60 days was far too long time, there was a need for finding out a better solution to get all the data needed from IMDb. Wget is a good program but aria2c is even better.

2.3 aria2c

Aria2 is a lightweight multi-protocol and multi-source command-line download utility [5]. Wget downloads one file at a time and after downloading, there is a need to run another wget to download another file. Aria2c works differently.

Aria2c has a queue of files. It downloads a file and after that, it downloads the next file on the list. With aria2c the user can see how many files have been downloaded and how many files are left. The total amount of data was 2,519,846 files which took over 200 GB. With six instances of aria2c were running, it took only 7 days to download all the data. Based on this, aria2c seems to be roughly 40% faster than wget in a large number of files. Sanjeev has tested which is the best way to download files and aria2c was the fastest [6].

2.4 sed

Sed is a stream editor that can be controlled with commands. A stream editor is used to perform basic text transformations on an input stream [7]. When there are multiple files which have the same information, it is easy to make a script that manipulates every file. Aria2c downloaded HTML files which had a huge amount of unneeded information. With multiple *sed* commands, it was possible to extract unneeded information. HTML files are pure text files which the browser reads and uses HTML commands for showing commands as a webpage. Editing an HTML file with a text editor shows the HTML file as a text. With a text editor, it could be possible to delete all the data which is not needed but when there are a huge amount of files, it is better to do a script that manipulates all the text files automatically.

IMDb and Steam generate HTML files automatically. Automatically generated HTML files have the same data structure. With the same data structure, it is possible to write scripts that remove the same text in every file. Sed is quite efficient for text manipulation.

2.5 Bash Shell Script

Shell is a program that takes user commands and gives them to the operating system.

Bash Shell script is a computer program written in the Bash programming language. A script can be programmed to run multiple commands. In this thesis, there were a huge amount of files that needed manipulating with multiple different commands. Instead of manipulating files one by one, script automated manipulating of files. After figuring out what commands were needed, the script uses those commands to every file.

2.6 screen

“The screening program allows you to use multiple windows (virtual VT100 terminals) in Unix.”[8]. A screen allows programs and scripts to run on the background. The screen makes running programs and script, which takes a long time, much easier to use. With a screen, the user can run a program and come later to see the result. Without a screen, the user must be connected to the server with SSH and the connection must be up while the program or script is running. When running a screen, the user can either run multiple screens or open multiple windows on the screen and run multiple shells. Running multiple shells simultaneously has an advantage when using a processor that has multiple cores. Some programs can only use one core at a time and to get full advantage of a processor that has six cores, it needs to run six instances of the program.

2.7 MySQL

“MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.” [9]. Databases are made for storing and comparing data. SQL stands for a Structured Query Language. After storing data to the database, the user can make different kinds of queries for analyzing data. With a huge amount of data, MySQL might not be the best possible database to use because even simple queries can take hours to compile. On the database made for this thesis, there were 166,031 movie tags and 95,654 game tags. When making a query where movie tags and game tags are compared together, MySQL makes 166,031 times 95,654 lines. The total number of lines is 15,881,529,274. After that, MySQL checks which lines fit in the query and show them. The table that has basic information from movies has 5,388,482 lines and when connecting that table to a movie tag table, MySQL makes a temporary table that has 894,655,054,942 lines. Making queries from 894 billion lines takes a huge amount of resources.

For this thesis, there was no need for all the movie data. With MySQL, there is one query that makes new smaller tables. With that command, movie data could be separated for 10 smaller tables which had movie data separated by years. For this thesis, there was a need for analyzing movie data year by year, between 2008 and 2018.

3 WORK

To compare movie tags and game tags, there is a need for a database. Setting up the database, adding data and making database queries are an important part of this thesis.

3.1 Setting up server

A server is a computer that has remote access. This whole thesis could be done with a personal computer but there are some benefits that a server offers. When there are some calculations or data handling that needs a lot of time, it is better to set up a server and let the server do the work. With a personal computer, there is a risk that the computer crashes and all the work that has been done must be done again. Another disadvantage is that the computer shares its resources between programs so that calculations take more time and using a computer is slower while it is doing work.

There are three ways of getting a server: buying a server, renting a dedicated server and renting a virtual private server. The difference between a dedicated server and VPS is that on the dedicated server the user has an own machine and all the power of the machine can be used by the user. On VPS there is one physical server which is shared with users. Every user has a maximum amount of resources which they can use. VPS is a cheaper option for a dedicated server because the company does not need to have all the actual hardware which they are renting.

There are a lot of companies that offer VPS. Comparing the offers of the companies is not easy because the price is not the only thing to compare. Computing power, the amount of memory, the size of a hard disk and the speed of an internet connection differs a lot and most companies have more than 1 different kind of package. To make comparing even more difficult, some companies have a fast internet connection, but they slow it down after the user has moved a certain amount of data per month. Before getting VPS, it is good to read some user reviews from companies to know exactly what you are paying for and is it good. For this thesis, VPS was the best solution and the author acquired it from Contabo.

Setting up VPS is easy. The user chooses an operating system from the webpage and the company will automatically install the selected operating system with selected add-ons. Choosing and installing the operating system takes a couple of minutes. The VPS company sends an email which has the server's login information and IP after they have installed the operating system. Linux has several basic programs installed but not all the programs needed for this thesis. Debian Linux has an apt-get system which installs programs. There are different commands on different Linux distributions but all of them have an easy way to install programs. Some programs, like MySQL, also need some setting up during installation and there are a lot of webpages that will guide through installation.

3.2 Downloading data

3.2.1 IMDb data

For this thesis, important information about movies was the year it was published and what tags users had given to it. IMDb shares a lot of data from movies via tsv files. Tsv files do not have user tags on them, only basic information about movies, such as crew, title and when it was made.

Every title on IMDb has a page that shows what tags users have added to that title. The only way to get user tags is to download all user tag pages. There are some sites, programs and code examples of how to get information from IMDb more easily. None of them downloads user tags. User tags can be found via URL

https://www.imdb.com/title/<movieID>/keywords?ref_=tt_stry_kw.

On 1.7.2018 the latest title on IMDb has an ID number 8,773,796.

When downloading a lot of material from the web site, there is a risk that the web server will ban an IP-address. Banning the IP-address is one way to block DoS-attacks. On DoS-attacks crackers use multiple computers to make a large amount of traffic to the website. Eventually, there is so much traffic that the web server cannot give all the data asked. If there is a large amount of traffic from one IP address, webservers may block that IP address, so it can handle the rest of the traffic which is not suspicious. Testing if it is possible to download several files from a webpage can be done with a script (Figure 1). To be sure that the script will work and it will download everything, there is a need for monitoring results. Monitoring can be done by opening another screen and typing `ls | wc -l`. That will count how many files are in the folder. Running that command a couple of times, the number of files should be growing. The amount is not the only thing to monitor. The amount of files can be growing even if the server has blocked the IP-address. In that case, the script downloads files, but instead of right data, there is an only notification that the IP-address has been blocked. Opening some files with *pico* tells what is inside the files. If there is not anything about a block and the data seems to be right, the script is working and the server has not blocked the IP-address.

Counting the amount of files, waiting for 1 minute and counting the amount of files again, gives the rough estimation of how many files will be downloaded per minute. If the total amount of files is known, this estimation can be used to calculate how much time it will take to download everything.

IMDb does not block the IP-address when downloading several files. The server allows the script download needed pages. With the script that is shown in Figure 1, it would take about 60 days to download everything needed for this thesis.

Monitoring the use of a processor with a command *top* tells that the script uses only a 1/6 of the processor power. The reason that the script does not use all the available computing power is that one instance of *wget* cannot use multiple cores and with a 6-core processor that means that only 1/6 of the computing power is used. Small download managers are simple programs. There is not any small and free program for downloading which handles multiple cores. There is a better solution for downloading multiple files than wget. Aria2c is more efficient. Aria2c cannot use all the cores. For using all cores, the server must run multiple instances of aria2c. When the amount of files is known, sharing the work amount for all cores can be done. At the beginning of June, the latest title on IMDb had an ID number 8,773,796. Dividing that by 6 gives a number which tells how many titles each script will download. The first script downloaded files from 1 to 1,462,299, the second script downloaded files from 1,462,300 to 2,924,599, the third script downloaded files from 2,924,600 to 4,386,898, the fourth script downloaded files from 4,386,899 to 5,849,197, the fifth script downloaded files from 5,849,198 to 7,311,497 and the last script downloaded files from 7,311,498 to 8,773,796. Aria2c has the ability to skip files that do not exist. The last movie had an identification number 8,773,796 and still, there were only 5,388,482 titles in IMDb. That means that there have been some entries that IMDb has deleted. The used aria2c script is shown in Figure 2.

```
#!/bin/bash
for i in {1.. 10}
do
  wget https://www.imdb.com/title/tt000000$i/keywords?ref_=tt_stry_kw
done
for i in {10.. 100}
do
  wget https://www.imdb.com/title/tt00000$i/keywords?ref_=tt_stry_kw
done
for i in {100.. 1000}
do
  wget https://www.imdb.com/title/tt0000$i/keywords?ref_=tt_stry_kw
done
for i in {1000.. 10000}
do
  wget https://www.imdb.com/title/tt000$i/keywords?ref_=tt_stry_kw
done
for i in {10000.. 100000}
do
  wget https://www.imdb.com/title/tt00$i/keywords?ref_=tt_stry_kw
done
for i in {100000.. 1000000}
do
  wget https://www.imdb.com/title/tt0$i/keywords?ref_=tt_stry_kw
done
for i in {10000000.. 8773796}
do
  wget https://www.imdb.com/title/tt$i/keywords?ref_=tt_stry_kw
done
```

FIGURE 1. Script for testing downloading user tags from IMDb

```
#!/bin/bash
for i in {1.. 10}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt000000$i/keywords?ref_=tt_stry_kw
done
for i in {10.. 100}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt00000$i/keywords?ref_=tt_stry_kw
done
for i in {100.. 1000}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt0000$i/keywords?ref_=tt_stry_kw
done
for i in {1000.. 10000}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt000$i/keywords?ref_=tt_stry_kw
done
for i in {10000.. 100000}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt00$i/keywords?ref_=tt_stry_kw
done
for i in {100000.. 1000000}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt0$i/keywords?ref_=tt_stry_kw
done
for i in {10000000.. 8773796}
do
  aria2c --file-allocation=none -c -x 10 -s 10 -d "imdb" -o $i.txt https://www.imdb.com/title/tt$i/keywords?ref_=tt_stry_kw
done
```

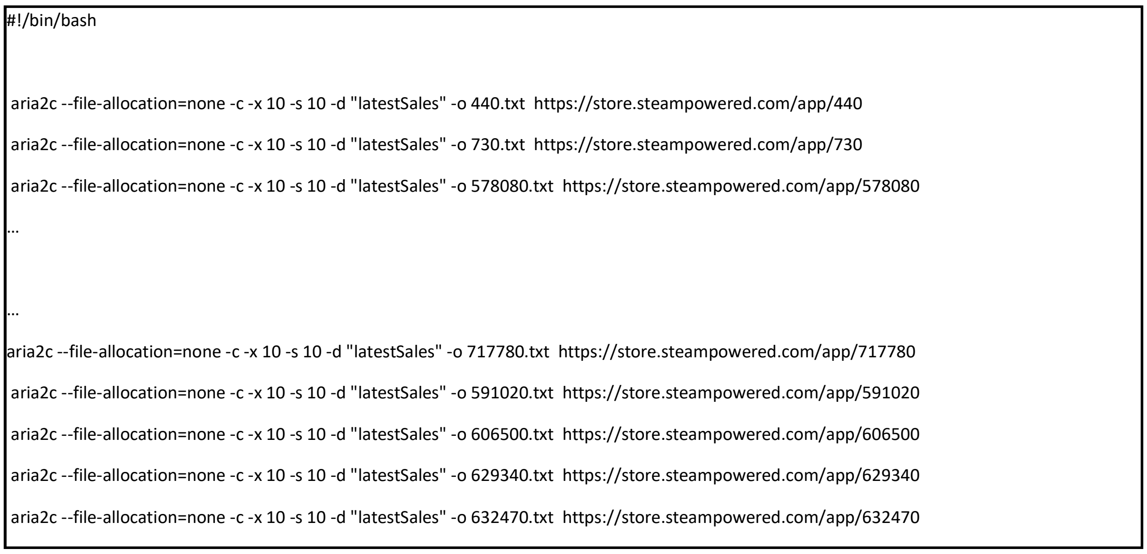
FIGURE 2. The script that downloads all the needed data from IMDb

3.2.2 Steam data

Ars Technica's information about Steam games sales has game names, sales and ID number of the games. With that ID number and name, it is possible to make a URL which leads to a game page in the Steam. URLs in the Steam are like

https://store.steampowered.com/app/<Game Id>

Generating a URL can be made with Excel. On a new cell adding text “https://store.steampowered.com/app/” and game ID after that makes a URL that directs to the game page. That gives an exact URL for 13,000 games. Adding an aria2c command before a URL gives a list of commands that will download all the data from Steam. When listing commands on a text file, Linux can run those commands. Sharing commands equally to six different files shares the workload quite equally to all cores. Some game pages have more data than others and it takes a little bit more time to download, but differences are quite small. All six scripts were as shown in Figure 3.



```
#!/bin/bash


 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 440.txt  https://store.steampowered.com/app/440
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 730.txt  https://store.steampowered.com/app/730
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 578080.txt  https://store.steampowered.com/app/578080
…


…
aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 717780.txt  https://store.steampowered.com/app/717780
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 591020.txt  https://store.steampowered.com/app/591020
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 606500.txt  https://store.steampowered.com/app/606500
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 629340.txt  https://store.steampowered.com/app/629340
 aria2c --file-allocation=none -c -x 10 -s 10 -d "latestSales" -o 632470.txt  https://store.steampowered.com/app/632470
```

FIGURE 3. The script that downloads all the needed data from Steam

3.3 Getting the information

Data stored on web servers is in an HTML format. That is a great format for browsers but it is not a good format for getting data to the MySQL table. On the HTML file, there is a lot of information that is only good for browsers. HTML is purely text and extracting information that goes to the database can be done with some text editing tools. For a better user experience, IMDb and Steam use automation for creating different pages. With automation, every page is in a similar format. Information differs but main things are same on almost every IMDb page and on almost every Steam page. For this thesis, the information needed from

Steam was a publisher, name, user tags, tags, and prices. On IMDb, there are only user tags.

3.3.1 IMDb

There are only user tags on IMDb files, so there is no need for separating data to multiple files. To make data manipulation more efficient, it is, however, useful to separate files so that multiple cores can work together and manipulating data is more efficient.

3.3.2 Steam

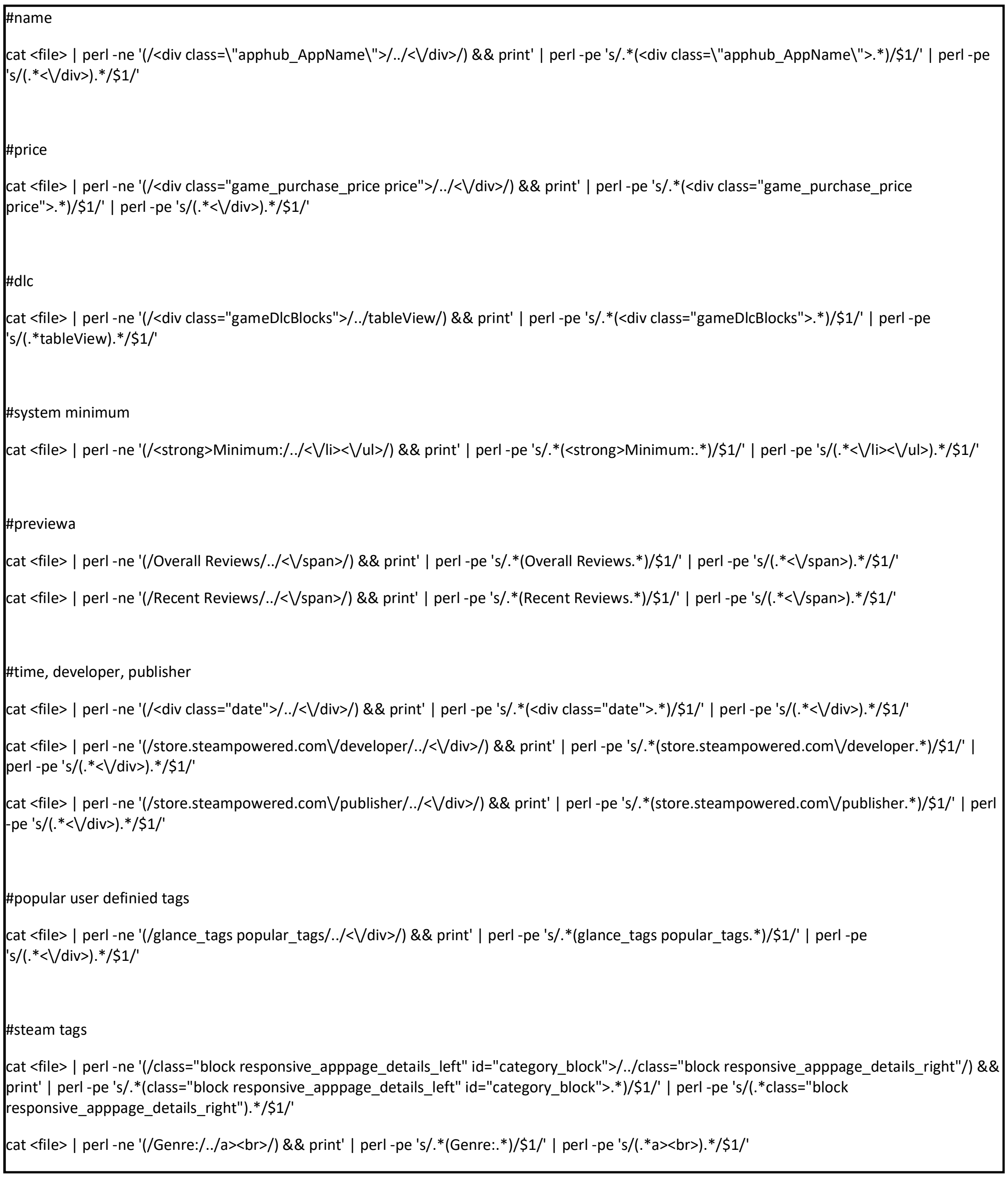
Steam files have more needed information than IMDb files. To get information into the database to different tables, it is easier to separate data to different files before. Information could be in the same file but it reduces the risk of error when data is in different files. Figure 4 shows a script that separates information into different files.

3.4 Deleting unnecessary text

HTML files have a lot of information that needs to be removed before the needed information is possible to insert into the database. Sed is a small simple program which can be used to remove text from text files in Linux.

3.4.1 IMDB

Most of the needed IMDb data comes directly from IMDb in the form that can be imported directly into the database. Only user-tag-files need modification. User tag pages have HTML codes in the files and data needed in this thesis is surrounded by HTML code. In IMDb files, there are movie ID, tags and the amount of how trustfully tags are. Everything else in those files is data that is not needed in this thesis. On IMDb, any user can add (almost) any tag to the movie. After the tag has been added, people can approve or disapprove a tag. With that election system, other people can see if a tag really is approved or not. People can, for example, see that Riot On! [10] has 17 Plot Keywords but none of them have been agreed or disagreed and Man without Past [11] has 145 Plot Keywords, some of them are voted as relevant while some of the keywords do not have any votes and some are voted as irrelevant. For cleaning files, several commands are needed. Sed can delete a string, everything before or after, string or a line (Figure 5).



```
#name

cat <file> | perl -ne '(/<div class=\"apphub_AppName\">/../<\/div>/) && print' | perl -pe 's/.*(<div class=\"apphub_AppName\">.*)/$1/' | perl -pe
's/(.*<\/div>).*/$1/'


#price

cat <file> | perl -ne '(/<div class="game_purchase_price price">/../<\/div>/) && print' | perl -pe 's/.*(<div class="game_purchase_price
price">.*)/$1/' | perl -pe 's/(.*<\/div>).*/$1/'


#dlc

cat <file> | perl -ne '(/<div class="gameDlcBlocks">/../tableView/) && print' | perl -pe 's/.*(<div class="gameDlcBlocks">.*)/$1/' | perl -pe
's/(.*tableView).*/$1/'


#system minimum

cat <file> | perl -ne '(/<strong>Minimum:/../<\/li><\/ul>/) && print' | perl -pe 's/.*(<strong>Minimum:.*)/$1/' | perl -pe 's/(.*<\/li><\/ul>).*/$1/'


#previewa

cat <file> | perl -ne '(/Overall Reviews/../<\/span>/) && print' | perl -pe 's/.*(Overall Reviews.*)/$1/' | perl -pe 's/(.*<\/span>).*/$1/'

cat <file> | perl -ne '(/Recent Reviews/../<\/span>/) && print' | perl -pe 's/.*(Recent Reviews.*)/$1/' | perl -pe 's/(.*<\/span>).*/$1/'


#time, developer, publisher

cat <file> | perl -ne '(/<div class="date">/../<\/div>/) && print' | perl -pe 's/.*(<div class="date">.*)/$1/' | perl -pe 's/(.*<\/div>).*/$1/'

cat <file> | perl -ne '(/store.steampowered.com\/developer/../<\/div>/) && print' | perl -pe 's/.*(store.steampowered.com\/developer.*)/$1/' |
perl -pe 's/(.*<\/div>).*/$1/'

cat <file> | perl -ne '(/store.steampowered.com\/publisher/../<\/div>/) && print' | perl -pe 's/.*(store.steampowered.com\/publisher.*)/$1/' | perl
-pe 's/(.*<\/div>).*/$1/'


#popular user definied tags

cat <file> | perl -ne '(/glance_tags popular_tags/../<\/div>/) && print' | perl -pe 's/.*(glance_tags popular_tags.*)/$1/' | perl -pe
's/(.*<\/div>).*/$1/'


#steam tags

cat <file> | perl -ne '(/class="block responsive_apppage_details_left" id="category_block">/../class="block responsive_apppage_details_right"/) &&
print' | perl -pe 's/.*(class="block responsive_apppage_details_left" id="category_block">.*)/$1/' | perl -pe 's/(.*class="block
responsive_apppage_details_right").*/$1/'

cat <file> | perl -ne '(/Genre:/../a><br>/) && print' | perl -pe 's/.*(Genre:.*)/$1/' | perl -pe 's/(.*a><br>).*/$1/'
```

FIGURE 4. The script that separates Steam information to different files

```
sed -i 's/<h1 class="header">Plot Keywords<\/h1>//g' title.txt
sed -i 's/<div id="keywords_content" class="header">//g' title.txt
sed -i 's/<div class="sort-controls">//g' title.txt
sed -i 's/Sort By: <select name="sort">//g' title.txt
sed -i 's/<option value="votes:descending" selected="selected">Relevance<\/option>//g' title.txt
sed -i 's/<option value="alpha:ascending">Alphabetical<\/option>//g' title.txt
sed -i 's/<\/select>//g' title.txt
sed -i 's/<span class="global-sprite lister-sort-reverse ascending" title="Descending order"><\/span>//g' title.txt
sed -i 's/<div class="header"><div class="nav"><div class="desc">//g' title.txt
sed -i 's/<table class="dataTable evenWidthTable2Col"><tbody>//g' title.txt
sed -i 's/<tr class="odd">//g' title.txt
sed -i 's/<\/div><\/div><\/div>//g' title.txt
sed -i 's/<div class="sodatext">//g' title.txt
sed -i 's/<div class="did-you-know-actions">//g' title.txt
sed -i 's/> Is this relevant?//g' title.txt
sed -i 's/<span>Relevant?<\/span>//g' title.txt
sed -i 's/<button class="cast-vote" value="up">Yes<\/button>//g' title.txt
sed -i 's/<button class="cast-vote" value="down">No<\/button>//g' title.txt
sed -i 's/<td><\/td><\/tr>//g' title.txt
sed -i 's/<\/tbody><\/table>//g' title.txt
sed -i 's/<div class="article" id="see_also//g' title.txt
sed -i 's/.*<\/a>/TAGI &/g' title.txt
sed -i ':a;N;$!ba;s/\n//g' title.txt
sed -i 's/ALKU/\nALKU/g' title.txt
sed -i 's/<\/div>/<\/div>\n/g' title.txt
sed -i 's/TAGI/\nTAGI/g' title.txt
sed -i 's/TAGI <\/a>//g' title.txt
sed -i 's/\t//g' title.txt
sed -i 's/  //g' title.txt
sed -i 's/<div class="sort-controls"><\/div>//g' title.txt
sed -i 's/<\/div>//g' title.txt
sed -i 's/<tr class="odd"><td class=.*//g' title.txt
sed -i 's/<span class="interesting-cast-vote" data-item-id=.*//g' title.txt
sed -i 's/<\/td><\/tr>//g' title.txt
sed -i 's/<\/td><td class="soda sodavote" data-item-votes.*//g' title.txt
sed -i 's/<\/a>//g' title.txt
sed -i 's/TAGI >/TAGI /g' title.txt
sed -i 's/\>/\n/g' title.txt
sed -i 's/\"> /\n/g' title.txt
```

FIGURE 5. The script that cleans IMDb files

3.4.2 Steam

Steam files have also several HTML commands. Getting information from those files differs from IMDb files because in Steam files most of the HTML code has gone when separating needed information to different files. Figure 6 shows commands that are needed to clean up Steam files.

3.5 Databases

When there is not a lot of data, data can be handled on an Excel. The amount of data handled in this thesis was so huge that Excel could not handle the amount. The database is the best choice for handling a lot of data. There are different databases. Because data was limited, there was not a need for looking for the best possible database for this thesis. The MySQL database is easy to use and that is the reason it has been used in this thesis.

3.5.1 Making the database

Loading data from files to the database can be done with one command in MySQL (Figure 7). In TSV files from IMDb, there are not any problems when adding information to tables. In data that has been extracted from HTML files, there is a problem because of the charset. IMDb lists movies all over the world. That is the reason that all the titles do not have a title in Latin alphabets. When using a database table that has Latin alphabets, there is a loss of information when adding a text that is wrote in different alphabets. Different alphabets are concerning only the name of the titles. There is a movie ID on the table so if needed, the movie title can be checked from the IMDb webpage. Steam has the same problem, but on the database, there is also a Steam page ID so the name of the game can be checked later from Steam if needed.

```
#editing publisher
sed -i 's/\t//g' publisher.txt                                                  #removes tabs
sed -i 's/.*\.publisher/BEGIN &/g' publisher.txt                                #add word BEGIN in front of every gameid
sed -i 's/<div class="date">//g' publisher.txt                                  #remove date div
sed -i 's/<div class="summary column">//g' publisher.txt summary                #remove colum
sed -i 's/<\/div>//g' publisher.txt                                             #remove /div
sed -i 's/<div class="summary column" id="developers_list">//g' publisher.txt   #remove id developers list
sed -i 's/,/\n/g' publisher.txt                                                 #replace , with enter
sed -i 's/<a href="https:\/\/store\.steampowered\.com\/search\/.*1_5_9__400">//' publisher.txt                   #remove search
sed -i 's/<a href="https:\/\/store\.steampowered\.com\/developer\/.*1_5_9__creator-home-product-page">//' publisher.txt #remove developer
sed -i 's/<a href="https:\/\/store\.steampowered\.com\/publisher\/.*1_5_9__creator-home-product-page">//' publisher.txt #remove publisher
sed -i 's/<a href="https:\/\/store\.steampowered\.com\/curator\/.*1_5_9__creator-home-product-page">//' publisher.txt #remove curator
sed -i '/https:\/\/store\.steampowered\.com\/search\//d' publisher.txt          #remove lines that has , in names
sed -i 's/<\/a>//g' publisher.txt                         #remove </a>
sed -i ':a;N;$!ba;s/\n//g' publisher.txt                  #remove entters
sed -i 's/BAGIN/\nBEGIN/g' publisher.txt                  #add enter before BEGIN


#editing names
sed -i 's/.*\.name/BEGIN &/g' name.txt                    #add word BEGIN in front of every gameid
sed -i 's/<div class="apphub_AppName">//g' name.txt       #remove apphub
sed -i 's/<\/div>//g' name.txt                            #remove <div
sed -i ':a;N;$!ba;s/\n//g' name.txt                       #remove enters
sed -i 's/BEGIN/\nBEGIN/g' name.txt                       #add enter before BEGIN
sed -i 's/.name/, /g' name.txt                            #replace word name with ,


#editing usertags
sed -i '/glance_tags popular_tags" data-appid/d'  usertags.txt                       #remove glance tags
sed -i '/<a href="https:\/\/store.steampowered.com\/tags\/en\//d'  usertags.txt      #remove steampowered
sed -i 's/\t\t\t.*//' usertags.txt2                                                  #delete everything after \t\t\t
sed -i '/ <a href/d' usertags.txt2                                                   #delete rows that has a href
sed -i 's/.usertagsadd/, /g' usertags.txt2                                           #replace usertagsadd with ,
sed -i '/usertags/d' usertags.txt2                                                   #remove word usertags


#editing tags
sed -i 's/.*\.tags/BEGIN &/g' tags.txt      #ad word BEGIN in front of every gameid
sed -i ':a;N;$!ba;s/\n//g' tags.txt         #remove enters
sed -i 's/BEGIN/\nBEGIN/g' tags.txt         #add enter before BEGIN
sed -i 's/<div/\n<div/g' tags.txt           #add enter before <div
```

FIGURE 6. Scripts that clean up files from Steam

```
#IMDb Tables

CREATE TABLE imdb_name_basics (nconst INT, primaryName VARCHAR(255), birthYear INT, deathYear INT, primaryProfession VARCHAR(255),
knownForTitles VARCHAR(255));

LOAD DATA INFILE '/var/lib/mysql-files/name.basics.tsv' INTO TABLE imdb_name_basics CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


CREATE TABLE imdb_basic (tconst INT, titleType VARCHAR(255), primaryTitle VARCHAR(255), originalTitle VARCHAR(255), isAdult INT, startYear INT,
endYear INT, runtimeMinutes INT, genres VARCHAR(255));

LOAD DATA INFILE '/var/lib/mysql-files/title.basics.tsv' INTO TABLE imdb_name_basics CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


CREATE TABLE imdb_crew (tconst INT, directors VARCHAR(255), writers VARCHAR(255));

LOAD DATA INFILE '/var/lib/mysql-files/title.crew.tsv' INTO TABLE imdb_crew CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES TERMINATED
BY '\n';


CREATE TABLE imdb_episode (tconst INT, parentTconst VARCHAR(255), seasonNumber INT, episodeNumber INT);

LOAD DATA INFILE '/var/lib/mysql-files/title.episode.tsv' INTO TABLE imdb_episode CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


CREATE TABLE imdb_ratings (tconst INT, averageRating DOUBLE, numVotes INT);

LOAD DATA INFILE '/var/lib/mysql-files/title.ratings.tsv' INTO TABLE imdb_ratings CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


CREATE TABLE imdb_principals (tconst INT, ordering INT, nconst VARCHAR(255), category VARCHAR(255), job VARCHAR(255), characters
VARCHAR(255));

LOAD DATA INFILE '/var/lib/mysql-files/title.principals.tsv' INTO TABLE imdb_principals CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


CREATE TABLE imdb_akas (titleId INT, ordering INT, title VARCHAR(255), region VARCHAR(255), language VARCHAR(255), types VARCHAR(255),
attributes VARCHAR(255), isOriginalTitle INT);

LOAD DATA INFILE '/var/lib/mysql-files/title.akas.tsv' INTO TABLE imdb_akas CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES TERMINATED
BY '\n';


CREATE TABLE imdb_tags (MovieID INT, tag VARCHAR(255), plus INT, all INT);

LOAD DATA INFILE '/var/lib/mysql-files/imdb_tags.txt' INTO TABLE imdb_ratings CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES
TERMINATED BY '\n';


#Steam tables

CREATE TABLE steam_games (name VARCHAR(255), ID INT, players INT, published DATE, price DOUBLE);

LOAD DATA INFILE '/var/lib/mysql-files/steam.txt' INTO TABLE steam_games CHARACTER SET UTF8 FIELDS TERMINATED BY '\t' LINES TERMINATED
BY '\n' (name, ID, players, @published, price) set published = STR_TO_DATE(@published, '%d.%m.%Y');


CREATE TABLE steam_tags (gameID INT, tag VARCHAR(255));

LOAD DATA INFILE '/var/lib/mysql-files/usertags.2valmis.txt' INTO TABLE steam_tags CHARACTER SET UTF8 FIELDS TERMINATED BY ';' LINES
TERMINATED BY '\n';
```

FIGURE 7. Command used for creating the database and adding data to it

4 RESULTS

4.1 50 Most used tags

On 50 most used tags in movies, there were 28 sex-related tags, 13 tags that were useless and only 9 tags that are useful. In games on 50 most used tags, there were 28 game related tags which do not apply to movies, 3 tags that are useless and 19 tags that are useful. There are 37,299 different tags on IMDb and only 350 different tags on Steam. Steam has a censorship on their tags. The difference comes from the reality that IMDb does not sell anything and Steam sells games. When only listing information, there is not so much need for a censorship than when selling. Steam will not allow a tag “junk” [12] but IMDb has that tag. That is only one example but the amount of different tags proves that IMDb allows more tags than Steam.

Another problem with tags is that some tags mean the same thing but they are spelled differently. On IMDb, there are 25 tags that have the word “zombie” in it. On Steam, there is only a “Zombies” tag. IMDb does not have a tag “Zombies”. That makes the comparison of tags much harder. In this thesis, Only the tags that are exactly the same are compared.

For a perfect result, IMDb tags should be changed to same as Steam tags or Steam games should be manually previewed and missing tags should be added.

4.2 Tags that found from both

There were 74 useful tags that can be found from IMDb and from Steam. There were 143 tags that can be found from IMDb and Steam but sometimes the same tag means a different thing in games and movies. For example, in survival movies, there is usually a big threat and main character(s) try to survive. In survival games, the player usually just kills enemies as long as possible. Survival games are usually casual games and survival movies are darker and made for the people who want to watch suspenseful movies. A split screen is another tag that cannot be compared between movies and games. In games, a split screen means that multiple players can play the game at the same time with one computer but it does not mean the same thing in movies. In movies, multiple people can always watch the same movie with one device so there is no need for the split screen in the same way that there is a need for a split screen in games.

On some tags, there were not enough movies or games that had that tag.

4.3 Correlation of tags

A correlation function on Excel tells whether a movie and game tags correlate. In Figure 8 there are graphs which tell how well they correlate in the same year and how well they correlate later. For this thesis, it is important to find out if the same tags trend at the same time or do movies or games begin the trend which will affect another, later on.

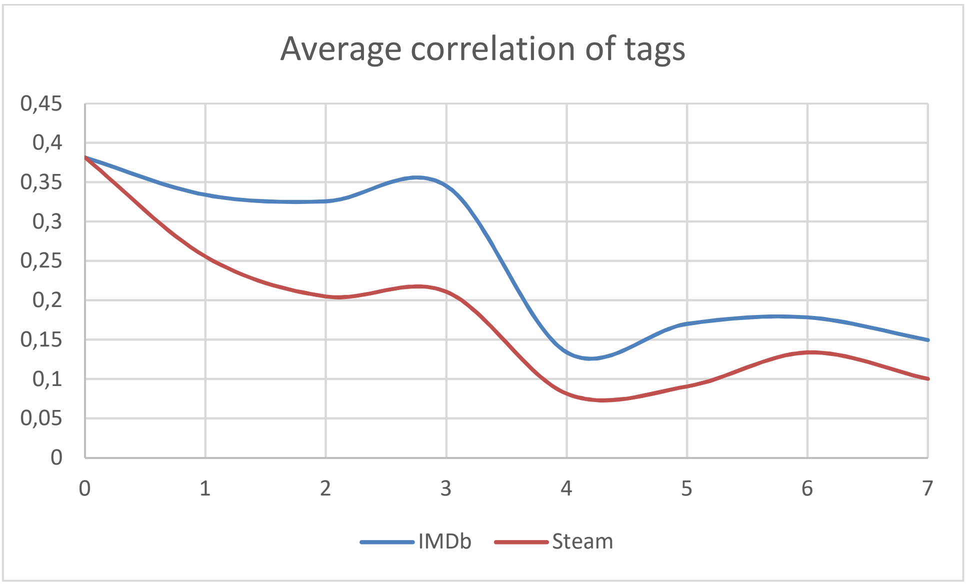


FIGURE 8. The average correlation of tags

The average correlation was the highest in the same year (Figure 8). These figures show that games follow more movies than movies follow games. Nowadays, it is relatively easy to make a game and publish it on Steam. The Cost of publishing a game on Steam is $100. This makes it very cheap to a group of students to make a game and publish it on Steam. Students might make a game just as a school project or as a hobby. On Steam, there are a lot of games that an amateur group has made and they have been made without a marketing research. Without a marketing research, game makers cannot know what will be the next trend and amateurs will more likely make a game based on what is trending right now. Making a game will take from a couple of weeks to a couple of years, depending on how complex, big and finalized the game will be. Most likely that is the reason why movie trends can be seen as game trends even 3 years after the trend has been popular.

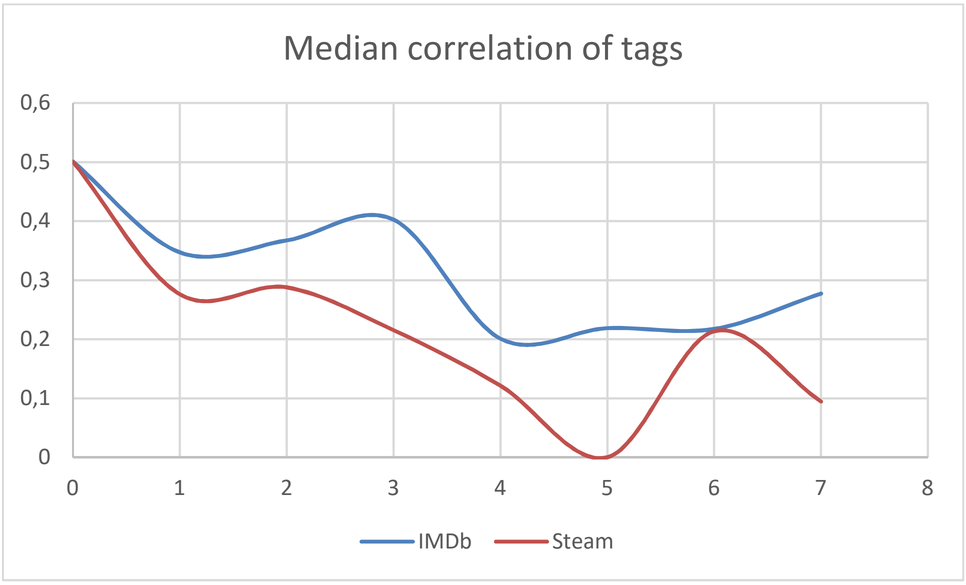


FIGURE 9. A median correlation of tags

A median correlation of tags is 0.5. A median means that half of the amount is higher than the median and half of them are lower than the median. When the median is 0.5 it means that 50% of the correlation of tags are higher than 0.5.

With median and average correlations, it is shown that game tags will follow movie tags. Without sales numbers, upcoming movies only tell what kinds of games will be published in a couple of years.

This study shows that there is a correlation between movie tags and game tags but it does not concentrate on reasons. Some correlations can be explained by current trends. For example, IMDb has a tag “F rated”. F rated means that the film is directed, written or starring by women. That tag has not existed before 2014 [13], but users have added that tag on movies later. There are most likely at least some movies that have been made before 2014 and that does not have an f rated tag and which would have an f rated tag if the movie would have been made today. F rated is only one example of how trends outside of the movie industry affect tags.

Another problem with movie tags is that only famous movies have a lot of tags and a lot of people have added or confirmed tags. Some movies have only one person who has added tags to them. When there is only one person who has added tags, and no one has confirmed tags, it is impossible to know if the tags are right.

The last problem with tags is that it is impossible to know if the tag is trending because of movies or because the tag itself is trending. Basically, if movies that were made 10 years ago would be made today, would they get the same tags, or would the tags be different? Most likely at least in some cases, movies would get different tags in different times. For example, at the time of cold war, people most likely saw politics in places that were not meant to be politic. People see more likely things that they consider in their everyday life.

4.4 Trends in movies 2008 - 2017

It is easy to see that there is not so much censorship on IMDb tags than there are on Steam tags. Because anyone can add almost any tag, there is many porn-related tags on IMDb. There are not almost any porn related tags on Steam, that is probably because Steam has been more conservative which kinds of games it allows to be released on Steam. Nowadays, Steam allows adult games. In this thesis, there have been only analyzed tags that can be found on both IMDb and Steam.

From those tags, there were clearly a couple of trends. In the year 2012 nudity became a trending tag. The highest value of nudity was in the year 2016 when there were 102 movies that had the tag nudity. After that, the popularity of nudity has decreased.

Another popular tag has been Blood. It was used 17 times in 2011 and 39 times in 2013. Blood has been popular since then but last year it dropped to 30. Blood tag is another tag that is currently decreasing.

The third biggest trend is the female protagonist. It has increased from the year 2012, when it was 15, to the year 2016, when it was 35. Last year it was 33. That makes it another tag that has most likely past a high point and is decreasing.

There are 7 tags that are currently rising: Destruction, Artificial Intelligence, Futuristic, Dark Comedy, Hacking, Mythology, Satire, Supernatural and Military.

4.5 Trends in games 2008 - 2017

There are 3 strong trends that have been rising lately: Anime, Female Protagonist and Retro. From those, only Retro is still rising. Other two have reached their peak point and they are coming less popular but they are still very strong. In 2017 there were 176 games that had a Female Protagonist tag, 188 games that had an Anime tag and 186 games that had a tag Retro.

There are 2 popular tags that have been rising from the year 2013 and have had their peak point at 2016: Space and Survival. In 2013 there were 20 games that had a tag survival and 15 games that had a tag Space. In 2016 there were 126 games that had a tag Survival and 117 games that had a tag Space. In the year 2017, the amount of that tag did not increase but there were almost the same amount of games that had those tags.

There are 5 tags that peaked in 2015, then decreased and last year they made their new record. Those tags are Gore, Historical, Dark Humor, Destruction, and Military.

There are 3 tags that have lately been rising year after year. Those tags are Fighting, Futuristic and Flight. Fighting has been rising since the year 2011 from 1 to 53 in 2017. Futuristic has been rising from 2013 from 3 to a value 43 in 2017. Flight has been rising from 2014 from 5 to 33 in 2017.

4.6 Trends that are rising from movies and games 2008 - 2017

There are 3 tags that are rising in movies and games. Those tags are Destruction, Futuristic and Military. There was also Dark Comedy that is rising on movies and Dark Humor that is rising on games. Those tags are almost the same so there is a change that gamers and people who watch movies just use a different tag.

Based on tags that are rising on both categories, the futuristic game about military destruction with a dark humor would cover all rising tags for 2018. The Verge published a “THE 39 GAMES WE CAN’T WAIT TO PLAY IN 2018”[14] list on December 26, 2017. From those 39 games, at least 9 have a tag destruction, Futuristic or Military. Most of those games were made by big game companies. That proves that game trends on this list also cover big game studios, not only small indie game studios.

5 CONCLUSION

Games and movies are both consumer products that fill the need for entertainment. Because they are filling the same need, there is some correlation between movie and game themes. The Correlation between IMDb and Steam tags are a different thing. There is a difference which tags can be used on IMDb and on Steam and because they are not using the same range of tags, tags are different.

5.1 Future

In this thesis, there were not any sales figures involved. This thesis compares only the amount of games and movies that have certain tags. This thesis cannot be used to predict whether a certain kind of game will be popular. It can only be used to predict if there will be a lot of certain kinds of games based on upcoming movies. Most likely the amount of movies and games will correlate some way to the amount of paying customers but that is something that needs to be studied in the future.

One problem with this thesis was different tags. There are two ways to avoid that problem. One way is to write more tags on Steam games. There were only 13,000 games studied in this work, so it would be possible to go through all the game webpages and write tags. That would not be very accurate but it would be close enough.

Another way to avoid the tag problem would be by comparing all the tags. IMDb uses more tags than Steam and more than one IMDb tag is affecting Steam tags. Researching which IMDb tags affect which Steam tags, there would be a need for analyzing what tags affect each other. That would need a lot of calculation power and a lot of work.

REFERENCES

1) Smith, C. 2018. 30 Interesting Steam Stats and Facts(December 2018). https://expandedramblings.com/index.php/steam-statistics/. Date of retrieval 20.9.2018

2) Orland, K. 2018. Valve leaks Steam game player counts; we have the numbers. https://arstechnica.com/gaming/2018/07/steam-data-leak-reveals-precise-player-count-for-thousands-of-games/. Date of retrieval 15.6.2018

3) Nithin, VR; Pranav, M; Sarath Babu, PB; Lijiya, A. 2014. Predicting Movie Success Based on IMDb Data

4) Lash, M; Zhao, K. 2016. Early Predictions of Movie Success: the Who, What, and When of Profitability. The University of Iowa. Page 2.

page 2, The University of Iowa

5) Arch Linux 2018. aria2. https://wiki.archlinux.org/index.php/aria2 Date of retrieval 25.9.2018

6) Nithyanandam, S. 2017. Download speed: vagrant init vs wget vs aria2. http://www.sanjeevnandam.com/blog/download-opentablewin-2008r2-standard-amd64-nocm-for-vagranttestkitchen. Date of retrieval 1.10.2018

7) Free Software Foundation. 2018. sed, a stream editor. Available https://www.gnu.org/software/sed/manual/sed.html. Date of retrieval 15.10.2018

8) The Trustees of Indiana University. 2018. In Unix, what is screen, and how do I use it?. https://kb.iu.edu/d/acuy. Date of retrieval 20.10.2018

9) Oracle Corporation. 2018. MySQL 5.7 Reference Manual. https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. Date of retrieval 20.10.2018

10) IMDb.com. 2004. Riot On! https://www.imdb.com/title/tt0427783/keywords?ref_=lgn_tt_stry_kw. Date of retrieval 12.12.2018.

11) IMDb.com, Inc 2002. Mies vailla menneisyyttä. https://www.imdb.com/title/tt0311519/keywords?ref_=tt_stry_kw. Date of retrieval 12.12.2018.

12) nanenj. 2014. [Steam Tags] Unexpected censorship with certain words. https://www.reddit.com/r/Steam/comments/1y6d66/steam_tags_unexpected_censor-ship_with_certain/. Date of retrieval 12.12.2018.

13) Guardian News and Media Limited. 2017. F-rated: IMDb introduces classification system to highlight work by women. https://www.theguardian.com/film/2017/mar/07/f-rated-imdb-introduces-classification-system-to-highlight-work-by-women. Date of re-trieval 14.12.2018.

14) Webster, A. 2017. THE 39 GAMES WE CAN'T WAIT TO PLAY IN 2018. https://www.theverge.com/2017/12/26/16807910/most-anticipated-games-2018-ps4-xbox-nintendo. Date of retrieval 14.12.2018