



Building a videogame recommendation system from scratch based on user and game data

by Jorge González Gómez

Tutored by Alejandro Bellogín Kouki

Introduction

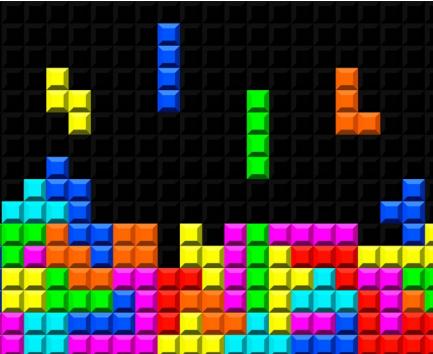
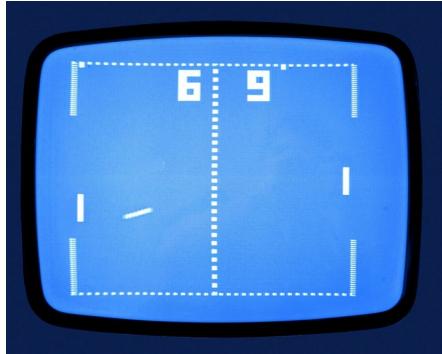
- What is a videogame / video game?
- What is a recommender system?
- The need for recommender systems in game distribution platforms.

Total slides: 52



What is a video game?

Examples of video games



What is a video game?

...but also examples of video games...



What is a video game?

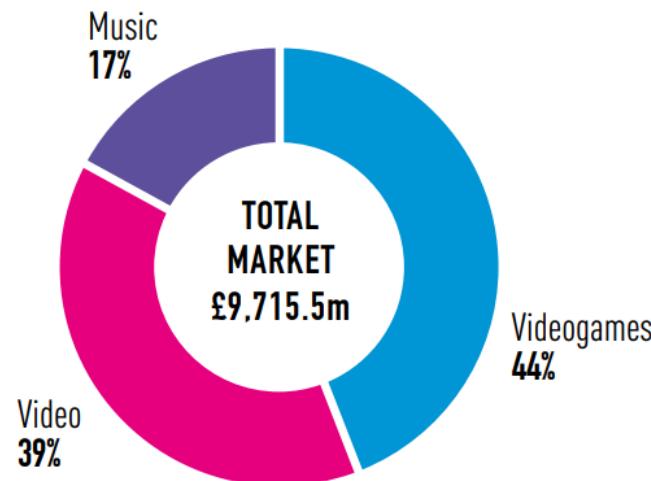
...and more examples of videogames



The video game industry

Digital Entertainment and Retail association revenue data

ERA ENTERTAINMENT
MONITOR: 2021 - (£M)



	ERA ENTERTAINMENT MONITOR 2021 - VALUE SALES (£m)			
	2019	2020	2021	change 20/21
Videogames	3,756.1	4,434.9	4,285.9	-3.4%
Video	2,610.6	3,311.7	3,752.3	13.3%
Music	1,453.7	1,543.6	1,677.3	8.7%
Total value	7,820.3	9,290.2	9,715.5	4.6%

What is Steam, and why?



What is Steam, and why?

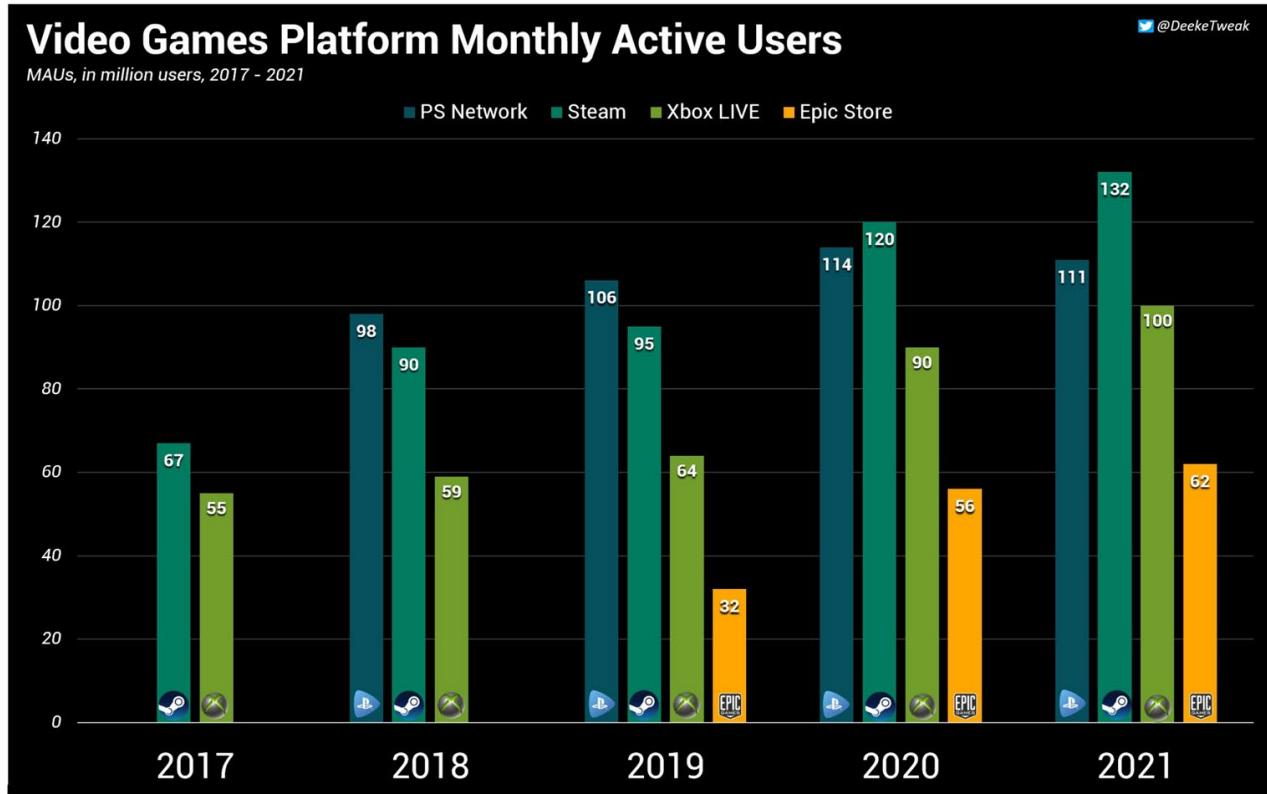


STEAMWORKS[™]

- ✓ Public API (Steamworks Web API)
- Currently the biggest user base
- Registers user playtime

The video game industry

Steam, PlayStation, Xbox and Epic Games



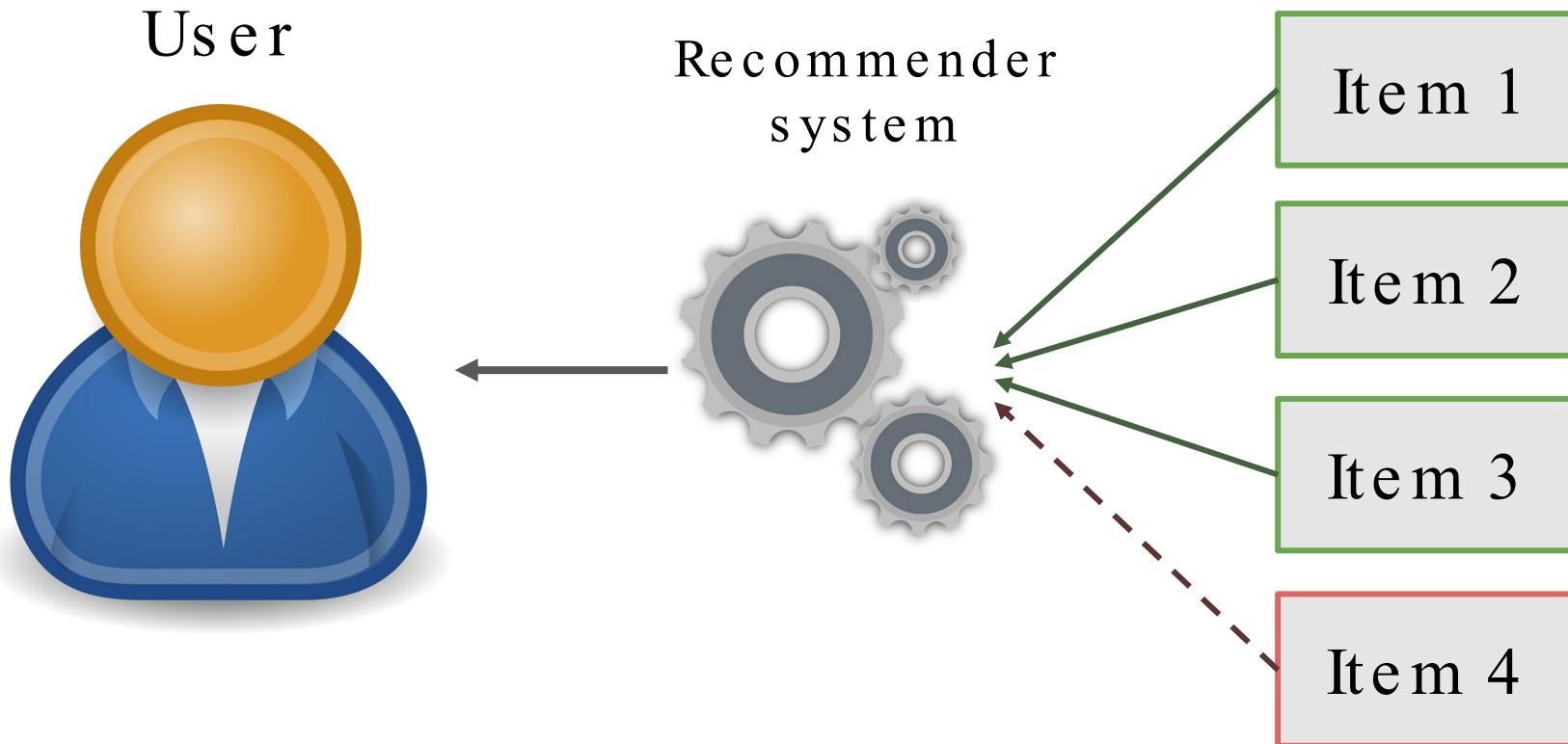
Steam kept growing after 2022



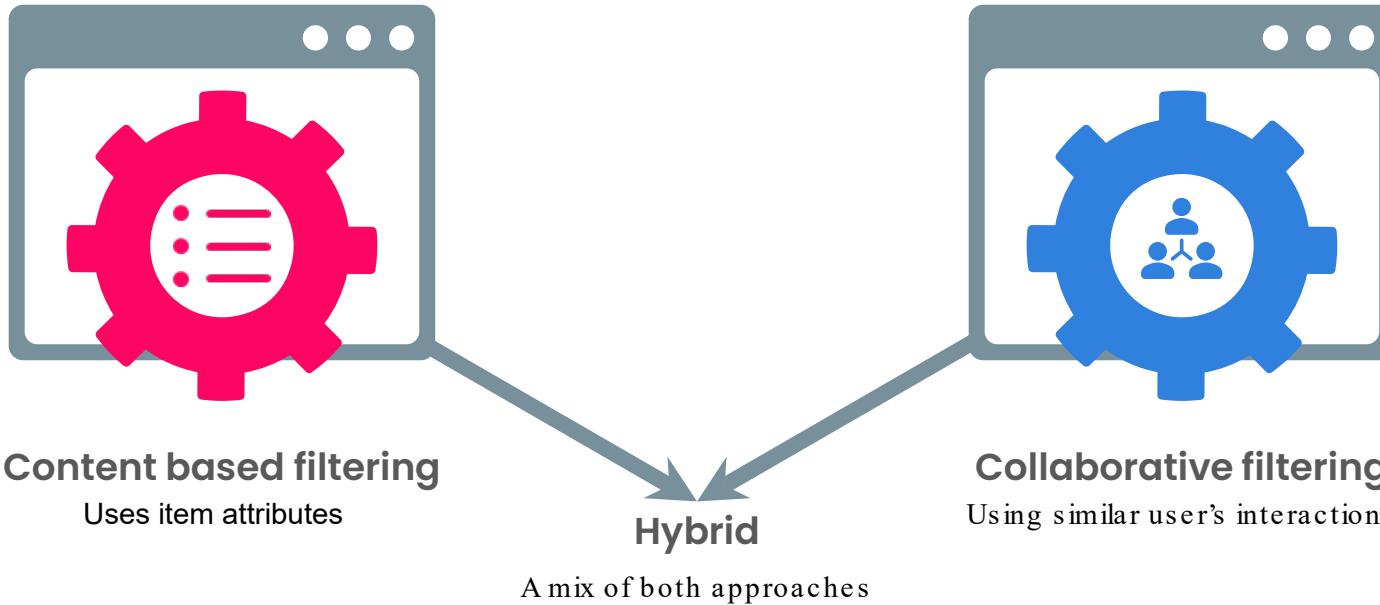
Nobody has enough time to check
all of these games out!

MONTH	GAMES THIS MONTH	MONTH	GAMES THIS MONTH
All 2022	10832	All 2023	4862
All 2021	10182	All 2016	4344
All 2020	9515	All 2015	2526
All 2018	8100	All 2014	1373
All 2019	7740	All 2017	6240
All 2016	4344	All 2013	2013
All 2015	2526	All 2012	1712
All 2014	1373	All 2011	1311
All 2013	2013	All 2010	1110
All 2012	1712	All 2009	1009
All 2011	1311	All 2008	808
All 2010	1110	All 2007	607
All 2009	1009	All 2006	406
All 2008	808	All 2005	205
All 2007	607	All 2004	104
All 2006	406	All 2003	63
All 2005	205	All 2002	32
All 2004	104	All 2001	18
All 2003	63	All 2000	10
All 2002	32	All 1999	8
All 2001	18	All 1998	5
All 2000	10	All 1997	3
All 1999	8	All 1996	2
All 1998	5	All 1995	1
All 1997	3	All 1994	1
All 1996	2	All 1993	1
All 1995	1	All 1992	1
All 1994	1	All 1991	1
All 1993	1	All 1990	1
All 1992	1	All 1989	1
All 1991	1	All 1988	1
All 1990	1	All 1987	1
All 1989	1	All 1986	1
All 1988	1	All 1985	1
All 1987	1	All 1984	1
All 1986	1	All 1983	1
All 1985	1	All 1982	1
All 1984	1	All 1981	1
All 1983	1	All 1980	1
All 1982	1	All 1979	1
All 1981	1	All 1978	1
All 1980	1	All 1977	1
All 1979	1	All 1976	1
All 1978	1	All 1975	1
All 1977	1	All 1974	1
All 1976	1	All 1973	1
All 1975	1	All 1972	1
All 1974	1	All 1971	1
All 1973	1	All 1970	1
All 1972	1	All 1969	1
All 1971	1	All 1968	1
All 1970	1	All 1967	1
All 1969	1	All 1966	1
All 1968	1	All 1965	1
All 1967	1	All 1964	1
All 1966	1	All 1963	1
All 1965	1	All 1962	1
All 1964	1	All 1961	1
All 1963	1	All 1960	1
All 1962	1	All 1959	1
All 1961	1	All 1958	1
All 1960	1	All 1957	1
All 1959	1	All 1956	1
All 1958	1	All 1955	1
All 1957	1	All 1954	1
All 1956	1	All 1953	1
All 1955	1	All 1952	1
All 1954	1	All 1951	1
All 1953	1	All 1950	1
All 1952	1	All 1949	1
All 1951	1	All 1948	1
All 1950	1	All 1947	1
All 1949	1	All 1946	1
All 1948	1	All 1945	1
All 1947	1	All 1944	1
All 1946	1	All 1943	1
All 1945	1	All 1942	1
All 1944	1	All 1941	1
All 1943	1	All 1940	1
All 1942	1	All 1939	1
All 1941	1	All 1938	1
All 1940	1	All 1937	1
All 1939	1	All 1936	1
All 1938	1	All 1935	1
All 1937	1	All 1934	1
All 1936	1	All 1933	1
All 1935	1	All 1932	1
All 1934	1	All 1931	1
All 1933	1	All 1930	1
All 1932	1	All 1929	1
All 1931	1	All 1928	1
All 1930	1	All 1927	1
All 1929	1	All 1926	1
All 1928	1	All 1925	1
All 1927	1	All 1924	1
All 1926	1	All 1923	1
All 1925	1	All 1922	1
All 1924	1	All 1921	1
All 1923	1	All 1920	1
All 1922	1	All 1919	1
All 1921	1	All 1918	1
All 1920	1	All 1917	1
All 1919	1	All 1916	1
All 1918	1	All 1915	1
All 1917	1	All 1914	1
All 1916	1	All 1913	1
All 1915	1	All 1912	1
All 1914	1	All 1911	1
All 1913	1	All 1910	1
All 1912	1	All 1909	1
All 1911	1	All 1908	1
All 1910	1	All 1907	1
All 1909	1	All 1906	1
All 1908	1	All 1905	1
All 1907	1	All 1904	1
All 1906	1	All 1903	1
All 1905	1	All 1902	1
All 1904	1	All 1901	1
All 1903	1	All 1900	1
All 1902	1	All 1899	1
All 1901	1	All 1898	1
All 1900	1	All 1897	1
All 1899	1	All 1896	1
All 1898	1	All 1895	1
All 1897	1	All 1894	1
All 1896	1	All 1893	1
All 1895	1	All 1892	1
All 1894	1	All 1891	1
All 1893	1	All 1890	1
All 1892	1	All 1889	1
All 1891	1	All 1888	1
All 1890	1	All 1887	1
All 1889	1	All 1886	1
All 1888	1	All 1885	1
All 1887	1	All 1884	1
All 1886	1	All 1883	1
All 1885	1	All 1882	1
All 1884	1	All 1881	1
All 1883	1	All 1880	1
All 1882	1	All 1879	1
All 1881	1	All 1878	1
All 1880	1	All 1877	1
All 1879	1	All 1876	1
All 1878	1	All 1875	1
All 1877	1	All 1874	1
All 1876	1	All 1873	1
All 1875	1	All 1872	1
All 1874	1	All 1871	1
All 1873	1	All 1870	1
All 1872	1	All 1869	1
All 1871	1	All 1868	1
All 1870	1	All 1867	1
All 1869	1	All 1866	1
All 1871	1	All 1865	1
All 1868	1	All 1864	1
All 1867	1	All 1863	1
All 1866	1	All 1862	1
All 1865	1	All 1861	1
All 1864	1	All 1860	1
All 1863	1	All 1859	1
All 1862	1	All 1858	1
All 1861	1	All 1857	1
All 1860	1	All 1856	1
All 1859	1	All 1855	1
All 1858	1	All 1854	1
All 1857	1	All 1853	1
All 1856	1	All 1852	1
All 1855	1	All 1851	1
All 1854	1	All 1850	1
All 1853	1	All 1849	1
All 1852	1	All 1848	1
All 1851	1	All 1847	1
All 1850	1	All 1846	1
All 1849	1	All 1845	1
All 1848	1	All 1844	1
All 1847	1	All 1843	1
All 1846	1	All 1842	1
All 1845	1	All 1841	1
All 1844	1	All 1840	1
All 1843	1	All 1839	1
All 1842	1	All 1838	1
All 1841	1	All 1837	1
All 1840	1	All 1836	1
All 1839	1	All 1835	1
All 1838	1	All 1834	1
All 1837	1	All 1833	1
All 1836	1	All 1832	1
All 1835	1	All 1831	1
All 1834	1	All 1830	1
All 1833	1	All 1829	1
All 1832	1	All 1828	1
All 1831	1	All 1827	1
All 1830	1	All 1826	1
All 1829	1	All 1825	1
All 1828	1	All 1824	1
All 1827	1	All 1823	1
All 1826	1	All 1822	1
All 1825	1	All 1821	1
All 1824	1	All 1820	1
All 1823	1	All 1819	1
All 1822	1	All 1818	1
All 1821	1	All 1817	1
All 1820	1	All 1816	1
All 1819	1	All 1815	1
All 1818	1	All 1814	1
All 1817	1	All 1813	1
All 1816	1	All 1812	1
All 1815	1	All 1811	1
All 1814	1	All 1810	1
All 1813	1	All 1809	1
All 1812	1	All 1808	1
All 1811	1	All 1807	1
All 1810	1	All 1806	1
All 1809	1	All 1805	1
All 1808	1	All 1804	1
All 1807	1	All 1803	1
All 1806	1	All 1802	1
All 1805	1	All 1801	1
All 1804	1	All 1800	1
All 1803	1	All 1799	1
All 1802	1	All 1798	1
All 1801	1	All 1797	1
All 1800	1	All 1796	1
All 1799	1	All 1795	1
All 1798	1	All 1794	1
All 1797	1	All 1793	1
All 1796	1	All 1792	1
All 1795	1	All 1791	1
All 1794	1	All 1790	1
All 1793	1	All 1789	1
All 1792	1	All 1788	1
All 1791	1	All 1787	1
All 1790	1	All 1786	1
All 1789	1	All 1785	1
All 1788	1	All 1784	1
All 1787	1	All 1783	1
All 1786	1	All 1782	1
All 1785	1	All 1781	1
All 1784	1	All 1780	1
All 1783	1	All 1779	1
All 1782	1	All 1778	1
All 1781	1	All 1777	1
All 1780	1	All 1776	1
All 1779	1	All 1775	1
All 1778	1	All 1774	1
All 1777	1	All 1773	1
All 1776	1	All 1772	1
All 1775	1	All 1771	1
All 1774	1	All 1770	1
All 1773	1	All 1769	1
All 1772	1	All 1768	1
All 1771	1	All 1767	1
All 1770	1	All 1766	1
All 1769	1	All 1765	1
All 1768	1	All 1764	1
All 1767	1	All 1763	1
All 1766	1	All 1762	1
All 1765	1	All 1761	1
All 1764	1	All 1760	1
All 1763	1	All 1759	1
All 1762	1	All 1758	1
All 1761	1	All 1757	1
All 1760	1	All 1756	1
All 1759	1	All 1755	1
All 1758	1	All 1754	1
All 1757	1	All 1753	1
All 1756	1	All 1752	1
All 1755	1	All 1751	1
All 1754	1	All 1750	1
All 1753	1	All 1749	1
All 1752	1	All 1748	1
All 1751	1	All 1747	1
All 1750	1	All 1746	1
All 1749	1	All 1745	1
All 1748	1	All 1744	1
All 1747	1	All 1743	1
All 1746	1	All 1742	1
All 1745	1	All 1741	1
All 1744	1	All 1740	1
All 1743	1	All 1739	1
All 1742	1	All 1738	1
All 1741	1	All 1737	1
All 1740	1	All 1736	1
All 1739	1	All 1735	1
All 1738	1	All 1734	1
All 1737	1	All 1733	1
All 1736	1	All 1732	1
All 1735	1	All 1731	1
All 1734	1	All 1730	1
All 1733	1	All 1729	1
All 1732	1	All 1728	1
All 1731	1	All 1727	1
All 1730	1	All 1726	1
All 1729	1	All 1725	1
All 1728	1	All 1724	1
All 1727	1	All 1723	1
All 1726	1	All 1722	1
All 1725	1	All 1721	1
All 1724	1	All 1720	1
All 1723	1	All 1719	1
All 1722	1	All 1718	1
All 1721	1	All 1717	1
All 1720	1	All 1716	1
All 1719	1	All 1715	1
All 1718	1	All 1714	1
All 1717	1	All 1713	1
All 1716	1	All 1712	1
All 1715	1	All 1711	1
All 1714	1	All 1710	1
All 1713	1	All 1709	1
All 1712	1	All 1708	1
All 1711	1	All 1707	1
All 1710	1	All 1706	1
All 1709	1	All 1705	1
All 1708	1	All 1704	1
All 1707	1	All 1703	1
All 1706	1	All 1702	1
All 1705	1	All 1701	1
All 1704	1	All 1700	1
All 1703	1	All 1699	1
All 1702	1	All 1698	1
All 1701	1	All 1697	1
All 1700	1	All 1696	1
All 1699	1	All 1695	1
All 1698	1	All 1694	1
All 1697	1	All 1693	

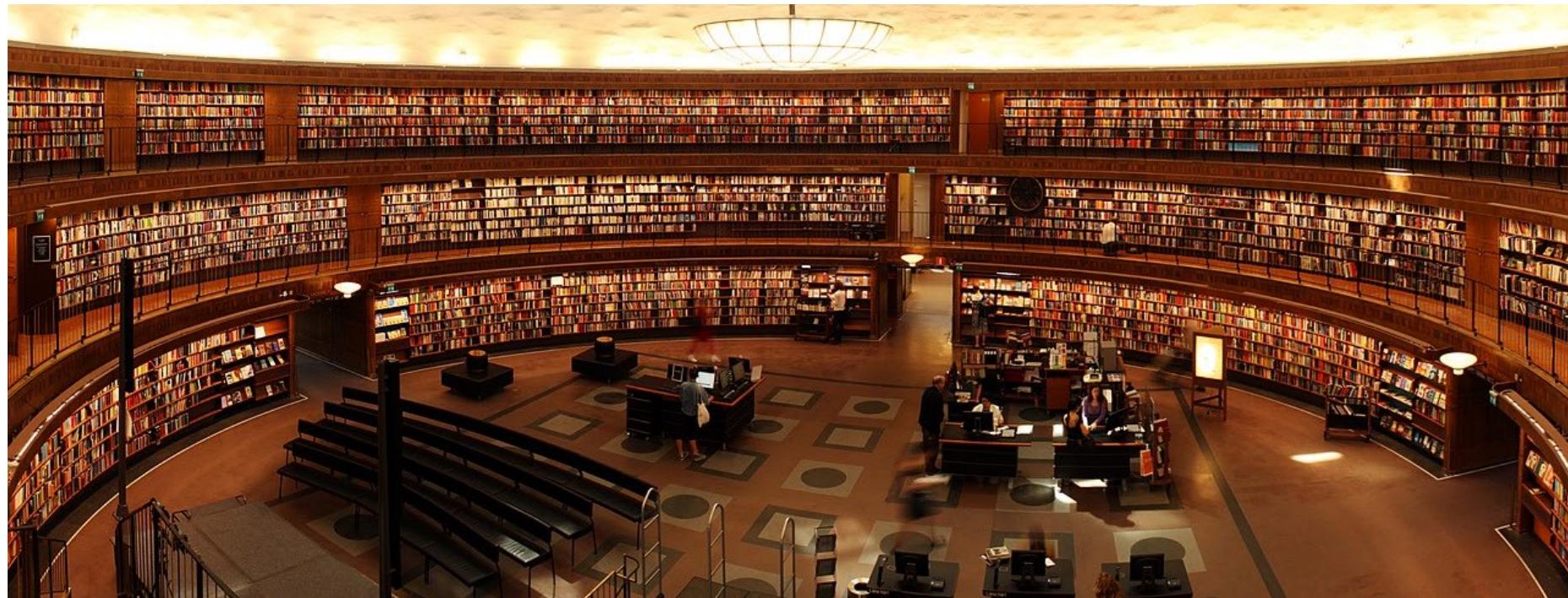
What is a recommender system?



Main types of recommender systems



Imagine entering a library...



...but none of these were ordered. But even if they are?

Tags only help you find a specific genre or theme

SPECIAL SECTIONS	GENRES			THEMES
Free to Play	Action	Role-Playing	Strategy	Anime
Demos	Arcade & Rhythm	Action RPG	Card & Board	Horror
Early Access	Fighting & Martial Arts	Adventure RPG	City & Settlement	Mystery & Detective
Steam Deck	First-Person Shooter	JRPG	Grand & 4X	Open World
Great on Deck	Hack & Slash	Party-Based	Military	Sci-Fi & Cyberpunk
Controller-Friendly	Platformer & Runner	Rogue-Like	Real-Time Strategy	Space
Remote Play	Third-Person Shooter	Strategy RPG	Tower Defense	Survival
VR Titles	shmup	Turn-Based	Turn-Based Strategy	PLAYER SUPPORT
VR Hardware	Adventure	Simulation	Sports & Racing	
Software	Adventure RPG	Building & Automation	All Sports	Co-Operative
Soundtracks	Casual	Dating	Fishing & Hunting	LAN
macOS	Hidden Object	Farming & Crafting	Individual Sports	Local & Party
SteamOS + Linux	Metroidvania	Hobby & Job	Racing	MMO
For PC Cafés	Puzzle	Life & Immersive	Racing Sim	Multiplayer
	Story-Rich	Sandbox & Physics	Sports Sim	Online Competitive
	Visual Novel	Space & Flight	Sports Sim	Singleplayer
			Team Sports	

Like an ordered library

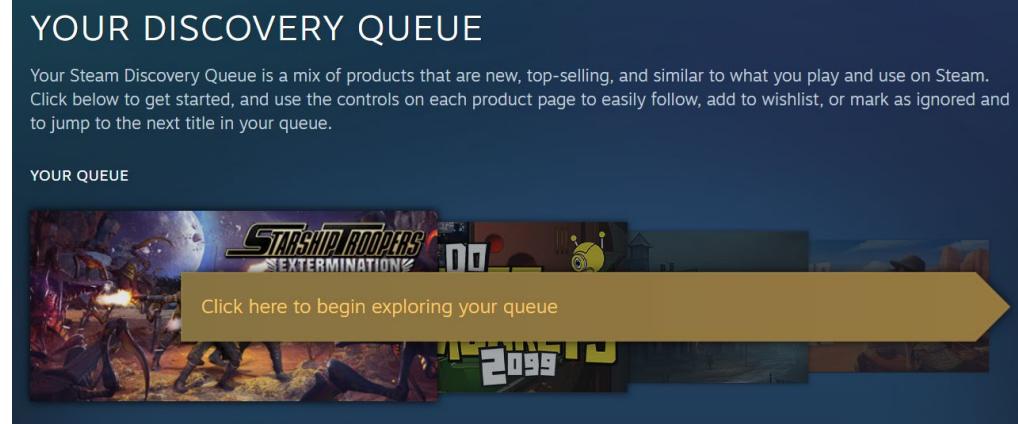
Good enough! **But we can do better.** Users need to:

- Manually check the genre/theme
- Unable to search for multiple tags from here
- Some are very vague (“Racing”, “Military”)

The need for recommender systems

- ▶ >68,000 games on Steam as of last month
- ▶ >800 new games per month
- ▶ Not enough time to check if you are interested in every game
- ▶ Steam has the Discovery Queue, but mostly shows popular games

Steam solved this issue, but...



Your Discovery Queue

Your Steam Discovery Queue is a powerful, new way of exploring the most popular new releases that you haven't yet seen. You can quickly browse through games that are suggested for you, and you can choose to follow the game, add it to your Wishlist, purchase it, or indicate that you are not interested. Your Discovery Queue is automatically refreshed each day with new, top-selling releases.



Steam interactive recommender

YOUR PLAYTIME



RogerFK
recent games
4464 hours total



54 hours
last played 43 minutes ago



17 hours
last played 1 hour and 24 minutes ago



29 hours
last played 17 hours ago



59 hours
last played 21 hours ago



21 hours
last played 1 day and 16 hours ago



423 hours
last played 2 weeks ago



19 hours
last played 3 weeks ago



39 hours
last played 5 weeks ago



15 hours
last played 6 weeks ago



41 hours
last played 7 weeks ago



35 hours
last played 7 weeks ago



66 hours
last played 7 weeks ago

RECOMMENDATIONS FOR YOU

Weight by popularity

POPULAR

NICHE

Filter by age

OLDER

NEWER

3 years

Add tag filters

Type a tag name

Add tag exclusions

Type a tag name

Exclude wishlist games

Save settings

PLAYERS LIKE YOU LOVE...

THE ONE WHO PULLS OUT THE SWORD WILL...

RELEASED ON MAR 28, 2022

CRAB GAME

RELEASED ON OCT 29, 2021

THE STANLEY PARABLE: ULTRA DELUXE

RELEASED ON APR 27, 2022

MILK OUTSIDE A BAG OF MILK OUTSIDE A BA...

RELEASED ON DEC 16, 2021

MILK INSIDE A BAG OF MILK INSIDE A BAG OF...

RELEASED ON AUG 26, 2020

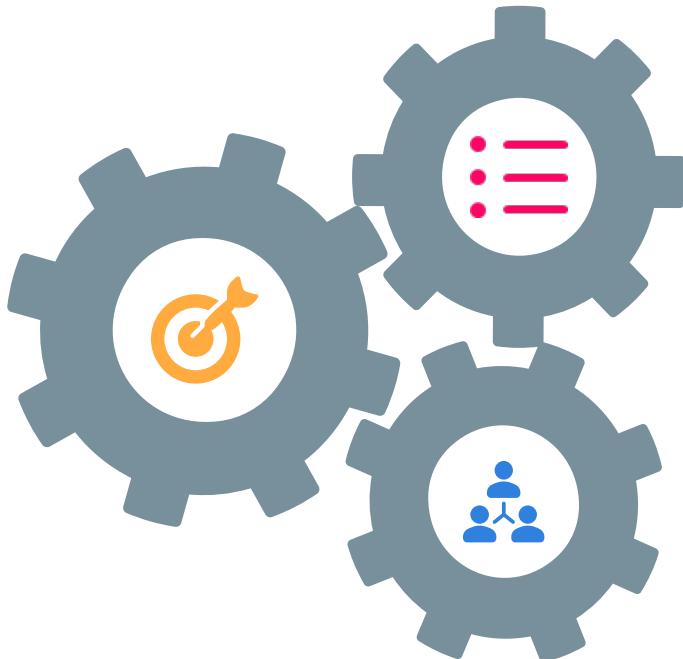
SUPERLIMINAL

RELEASED ON NOV 5, 2020



Our solution

A Videogame Recommender System



Concepts



User = player

Who we are trying to predict for.



Item = game

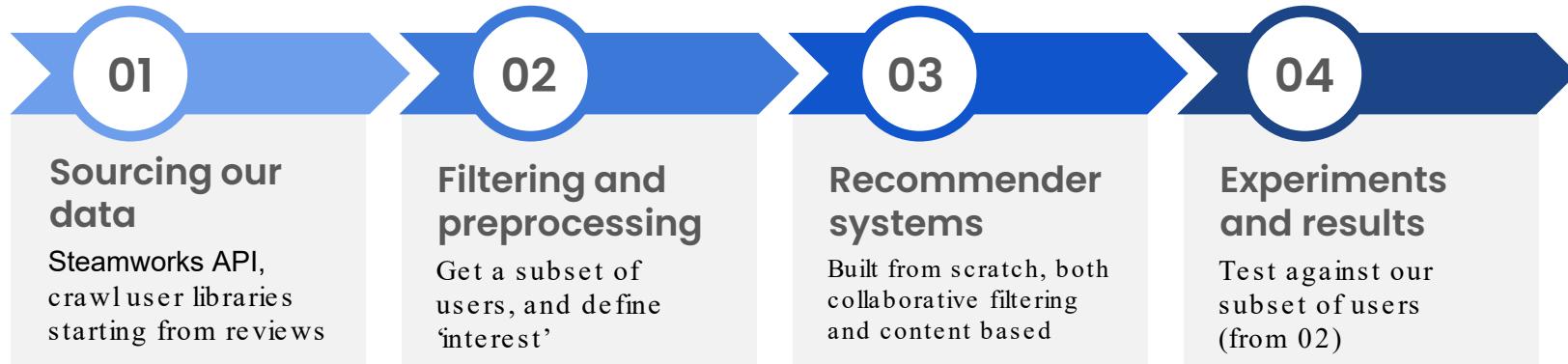
What we are trying to predict.



Playtime

The '*interest*' of a player in a game

Structure of the project



steamreviews



 **pandas**

datasketch

Sourcing our data

No way to get a list of Steam users directly.

Solutions:

1. Test random valid SteamIDs and see if they exist
(not time efficient)
2. Scrape online users from their website (might get banned from Steam / too slow)
3. Crawl reviews from selected games from their API (the most compliant, efficient way)

Sourcing our data

01

Manually select games

Pick popular games to get a good amount of different gamers.

02

Crawl all available reviews of the games

Get the users who reviewed these games

03

Store user and game data

User data includes their SteamID and the number of games owned

04

Gather user libraries

We can filter out private users and those with private game lists, including playtimes

But why not crawl the reviews of these users instead of using playtimes?

Why not crawl reviews of a user?

Unable to crawl all reviews of a particular user

but...

- Users do not leave reviews for all of their games
- “Troll” reviews

Examples of reviews in FIFA 23

208 people found this review helpful
129 people found this review funny  15

 Recommended
513.9 hrs on record

Posted: 18 March
i hate this game so much but i cant stop playing it

1,210 people found this review helpful
846 people found this review funny  34  7  20  65

 Not Recommended
35.6 hrs last two weeks / 348.4 hrs on record (1.4 hrs at review time)

Posted: 29 Sep, 2022 @ 5:42pm
Updated: 29 Sep, 2022 @ 7:10pm

Are you wasting my money again, son?



94 people found this review helpful
31 people found this review funny  5

 Recommended
728.3 hrs on record

Posted: April 28
this game chips my life away every minute
i play it (dont ever buy any fifa)

Why use playtimes



PROS

1. People have limited time: if they spend it in game A rather than game B, it hints they prefer it
2. We can get all playtimes, unlike all reviews
3. Not all games are reviewed, neither positively or negatively

CONS

1. You might get burned out after 500h in a game
2. Some people need 20 hours to find out if they dislike a game
3. Some games (e.g. Strategy/Simulation games) take a lot of time to play, but might not be as interesting to the user as smaller, shorter games (e.g. Adventure games)

Gathering item attributes

Content-Based recommender systems

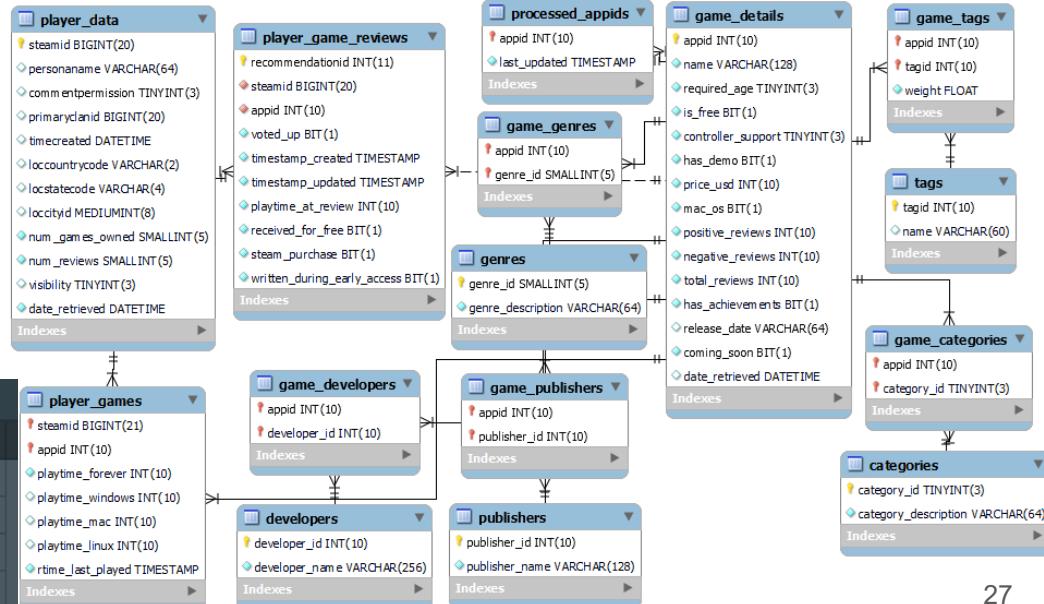
- Hypothesis: if a game is very similar to another, a user might be interested in it
- Steam provides information about games



Our data

steam_tfg.jgg	7.0 GiB
candidate_appids	4.6 MiB
categories	16.0 KiB
developers	4.5 MiB
game_categories	13.5 MiB
game_details	5.5 MiB
game_developers	5.5 MiB
game_genres	10.5 MiB
game_publishers	5.5 MiB
game_tags	10.5 MiB
genres	16.0 KiB
player_data	164.5 ...
player_games	274.6 ...
player_game_reviews	16.0 KiB
processed_appids	2.5 MiB
publishers	32.0 KiB
tags	6.5 GiB

steam_tfg.jgg.player_games: 50,045,823 rows total (approximately), limited to 1,000				
steamid	appid	playtime_forever	playtime_last_played	time_last_played
76,561,198,115,515,666	660,880	2,957,082	2023-03-29 22:20:...	
76,561,197,970,402,743	385,800	2,940,686	2023-04-11 20:00:...	
76,561,198,098,325,355	333,600	2,894,283	2023-04-05 18:02:...	
76,561,198,098,325,355	385,800	2,893,240	2023-04-05 18:02:...	
76,561,198,098,325,355	420,110	2,893,202	2023-04-05 18:02:...	



Tags and IDF: an hypothesis

IDF (Inverse Document Frequency)

IDF = In documents, IDF penalizes terms that are too frequent when ranking documents in search engines, like Google



Tags

User-defined,
ordered by
priority

Hypothesis:

“If a tag is too common, it might not be too useful for us”



Defining and normalizing “interest”

Formal explanation

Let u be a user (or player) and i be an item (or game).

Let us define the function $I(u, i)$ as the interest a user u has over the item i , which yields a value between 0 and 1, where 0 means no interest and 1 means maximum interest.

Then we can define “implicit interest” as the play time spent by a player u playing game i , normalized from 0 to 1, where $I(u, i) = 1$ (“the game the player is most interested in”) is the game with the highest playtime.

For example, if user u has three games i_1, i_2 and i_3 with playtimes of $I(u, i_1), I(u, i_2)$ and $I(u, i_3)$ of 100h, 50h and 20h respectively, then $I(u, i_1) = 1$ and $I(u, i_2), I(u, i_3)$ will be based on the user’s maximum playtime / interest, $I(u, i_1)$ in our case.

Different normalization techniques

$$x = \frac{x}{\max(I(u, i_n))}$$

$$\frac{\log(x)}{\log(\max(I(u, i_n)))}$$

$$\frac{\sqrt{x}}{\sqrt{\max(I(u, i_n))}}$$

Defining and normalizing “interest”

In other words...

We define the interest over a game, for each user, as the time they have spent playing that game relative to their most played game

Different normalizations techniques

Linear

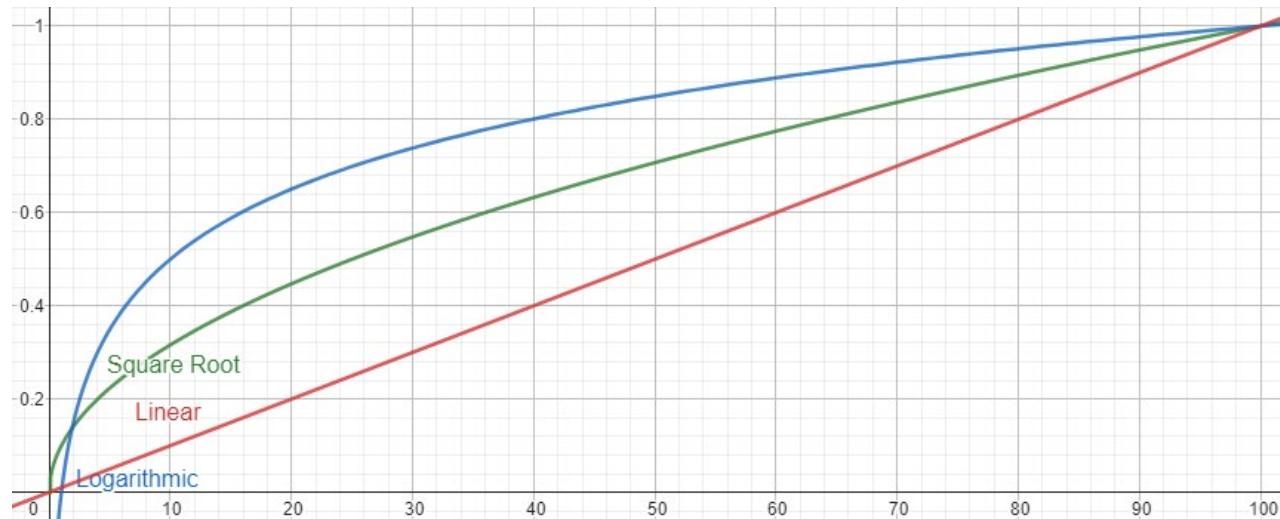
$$\frac{x}{\max(I(u, i_n))}$$

Logarithmic

$$\frac{\log(x)}{\log(\max(I(u, i_n)))}$$

Square Root

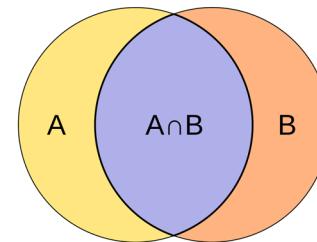
$$\frac{\sqrt{x}}{\sqrt{\max(I(u, i_n))}}$$



MinHash, LSH Ensemble and similarity

Jaccard similarity

$$\frac{|A \cap B|}{|A \cup B|}$$



“The number of items in common divided by the number of items of both sets”

Quick way of finding similar items: **MinHash LSH**, which approximates Jaccard and indexes those matches above a threshold t (saves all matches of Jaccard $> t$)

But Jaccard penalizes big sets, which might contain more information...

LSH Ensemble
by Erkang Zhu

$$\frac{|A \cap B|}{|A|}$$

Collaborative filtering (CF): relevant games

When MinHashing and using the LSH Ensemble, we can take top games

User X library	
ELDEN RING	100h
Counter-Strike	80h
Baldur's Gate	40h
RimWorld	10h
Subnautica	5h

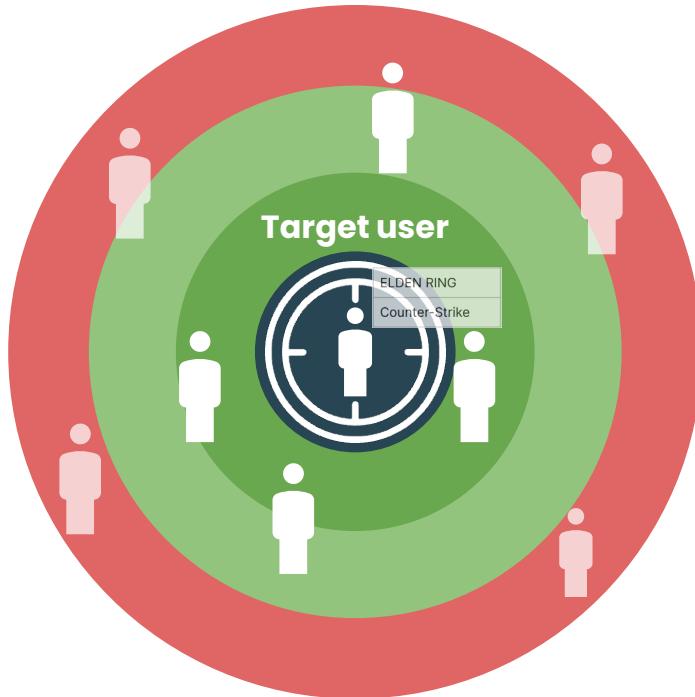
>60% of max playtime



User X MinHash
ELDEN RING
Counter-Strike

Collaborative filtering (CF): similarity

We want similar users to our target: LSH Ensemble to filter out unwanted users



Then apply one of these functions, taking into account every owned game where $R_{u,i}$ is the rating of a user over an item:

Raw:
$$\sum_{i \in I_u \cap I_v} R_{u,i} \cdot R_{v,i}$$
 "Just multiply ratings"

Cosine:
$$\frac{\sum_{i \in I_u \cap I_v} R_{u,i} \cdot R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} R_{v,i}^2}}$$
 "Take into account the total time spent in all games"

Pearson:
$$\frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u) \cdot (R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}$$
 "Same as before, but use average ratings into account"

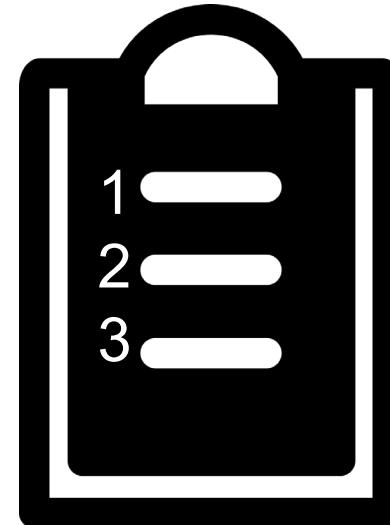
Remember: ratings = interest = playtime

Collaborative filtering (CF): scoring

N = number of games to recommend

K = number of similar users to use

1. Pick the top K similar users according to the selected method (which internally uses the LSH Ensemble)
2. Keep track of a dictionary **game -> score**
3. For **user** in all K similar users:
 - a. Get all games of **user**
 - i. $\text{game} += \text{interest}(\text{user}) \times \text{similarity}(\text{user}, \text{target})$
4. Order the list of games and return top N



2

Content-Based Filtering (CBF): finding games

Find the “preferred” game of the user

User Y library	
Action, RPG	100h
Horror, Action	80h
Racing, Sports	40h
Adventure, Indie	10h
Sports, Indie	5h



User Y attribute weight map	
Action	180h
RPG	100h
Horror	80h
Sports	45h
Racing	40h
Indie	15h
Adventure	10h

Content-Based filtering (CBF): similarity



Same as before, but we take item attributes (tags, genres, etc.) and tweak the LSH Ensemble threshold



Only with tags: To score tags we can adapt our CF, but instead of $R_{u,i}$ being the rating of a user over an item, it's item's weight of a tag:

Raw:
$$\sum_{i \in I_u \cap I_v} R_{u,i} \cdot R_{v,i}$$
 “Just multiply weights”

Cosine:
$$\frac{\sum_{i \in I_u \cap I_v} R_{u,i} \cdot R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} R_{v,i}^2}}$$
 “Penalize those with many tags”

Pearson:
$$\frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u) \cdot (R_{v,i} - \bar{R}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \bar{R}_u)^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (R_{v,i} - \bar{R}_v)^2}}$$
 “Penalize tags with low priority (bad!)”

Content-Based Filtering: scoring

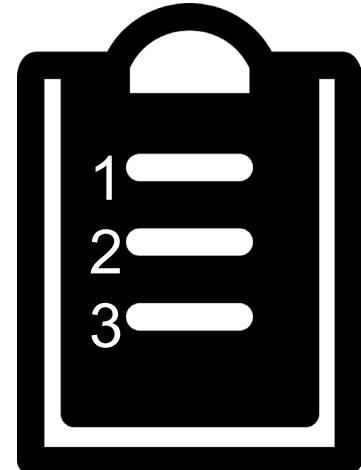
Preferred game	
Action	180h
RPG	100h
Horror	80h
Sports	45h
Racing	40h
Indie	15h
Advent.	10h

N = number of games to recommend

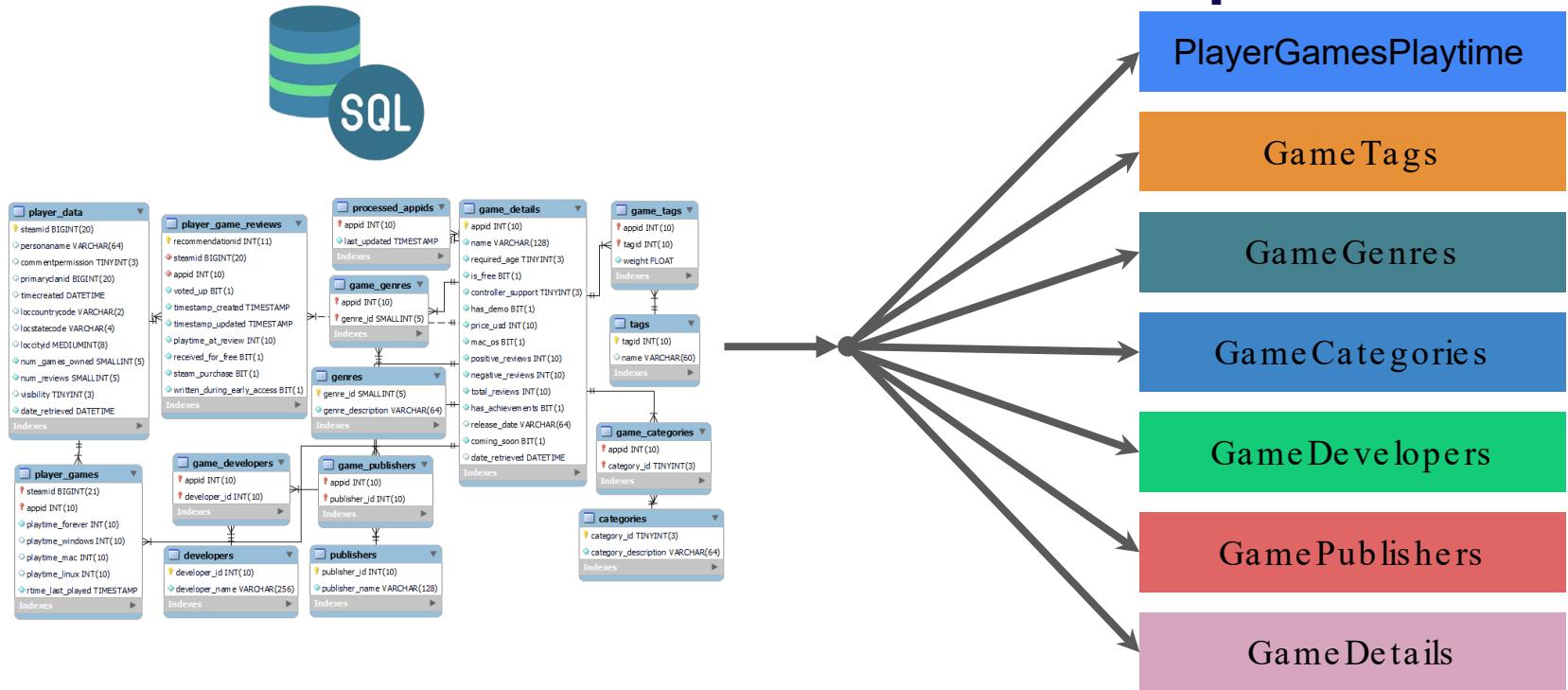
“Preferred game” = our item weight map

LSH threshold: t = “games above this approx. Jaccard similarity”

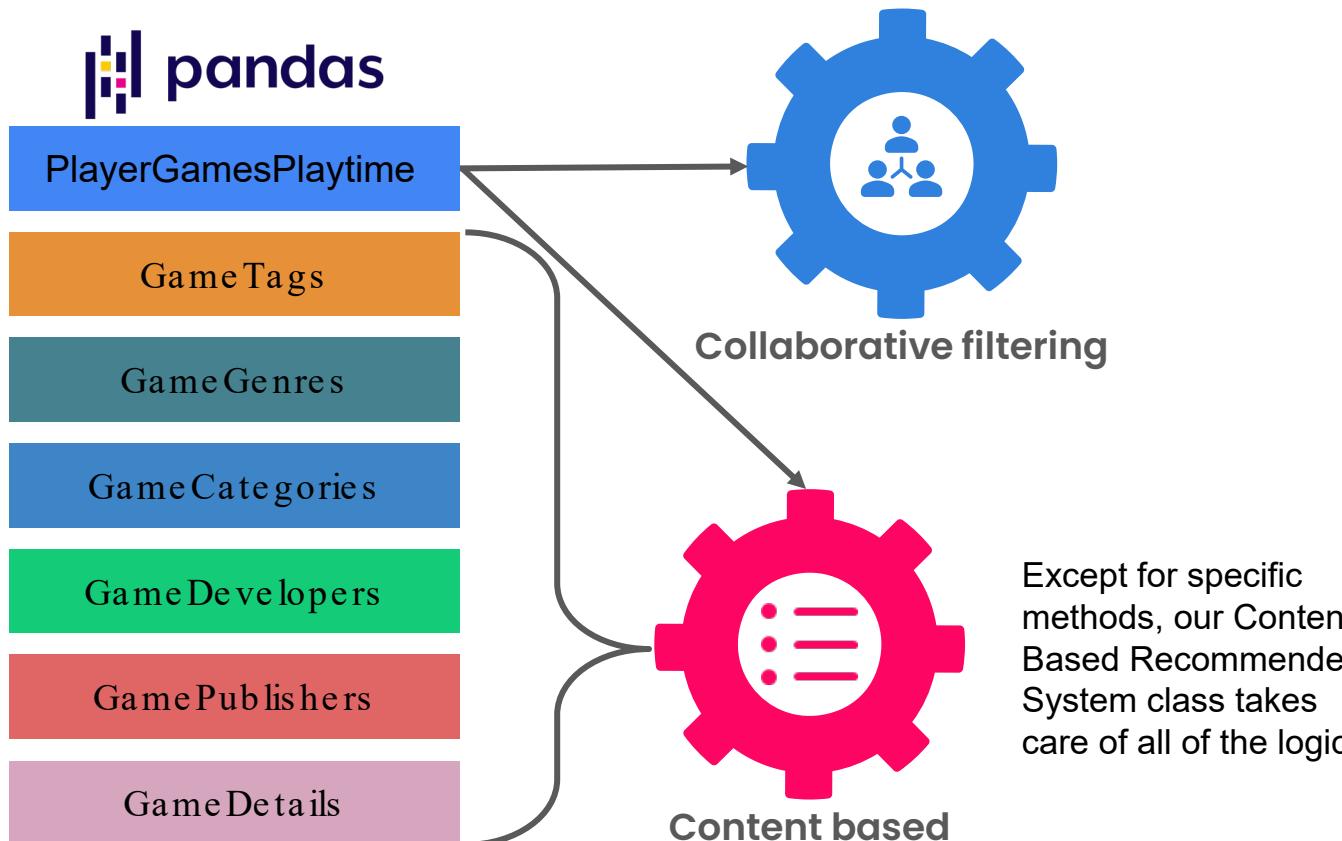
1. Pick the similar games above the LSH threshold t to our “preferred game”
2. For **game** in all K similar games:
 - a. **game** score = the real similarity to our “preferred game”
3. Order the list of games by their score and return top N



Structuring our data



Recommender systems



Experiments and results: Measuring accuracy



Split 80% / 20%

Our two ways to measure accuracy:
(Always at 5, 10 or 20 top results)



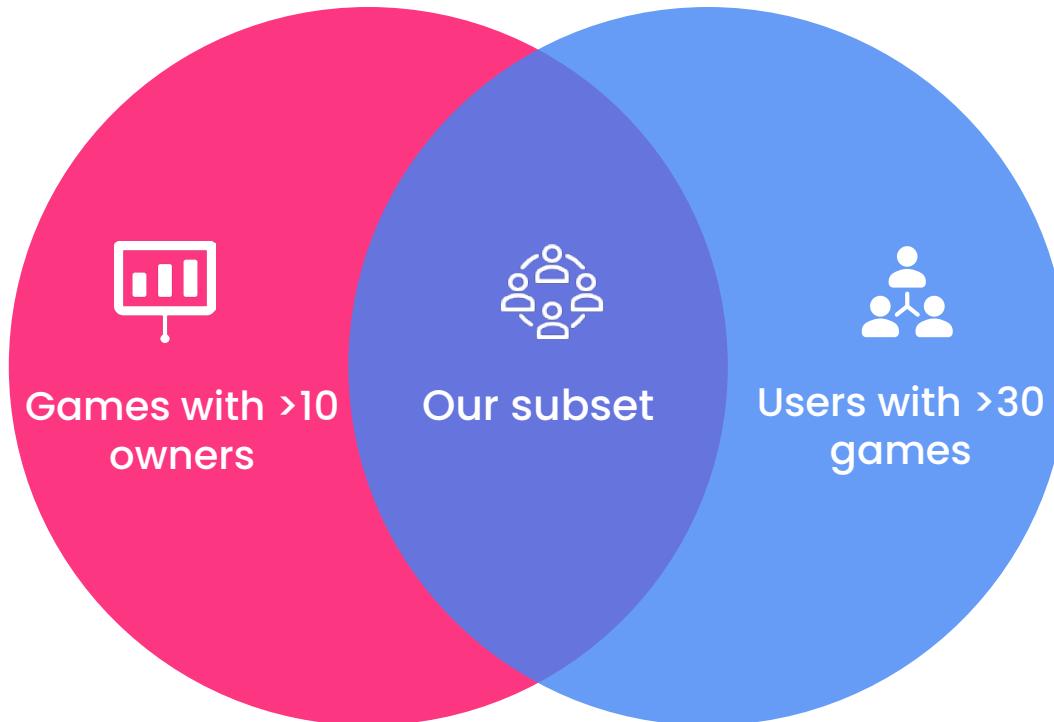
Precision $\frac{|\{Rel\} \cap \{Ret\}|}{|\{Ret\}|}$



Recall $\frac{|\{Rel\} \cap \{Ret\}|}{|\{Rel\}|}$

We will also measure time efficiency

Extracting a subset



Baseline

P@K = Precision at K. **R@K** = Recall at K

Combination	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)
Random	0.0046	0.0041	0.0042	0.0014	0.0025	0.0048	4.71
Top rated	0.0016	0.0010	0.0073	0.0005	0.0006	0.0087	705.54

Anything close or below 0.0046 in precision at 5 or below 0.0014 on recall at 5 means it performs worse than our random recommender

Our findings: Content Based Filtering

Top results for Content Based Filtering

Separated by each recommender system:

t means the LSH ensemble threshold

w_{idf} means the weight for our “IDF hypothesis”

Underscored highlights the best combination for each recommender system

Bold highlights the best overall for Content Based Filtering

Categories	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)	Raw Game Tags	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)
$t=0.42$	<u>0.0370</u>	<u>0.0304</u>	<u>0.0251</u>	<u>0.0114</u>	<u>0.0185</u>	<u>0.0302</u>	1869.48	$t=0.30, w_{idf}=0.60$	0.0472	<u>0.0382</u>	<u>0.0318</u>	0.0137	<u>0.0217</u>	<u>0.0364</u>	9379.58
$t=0.55$	0.0242	0.0150	0.0082	0.0076	0.0094	0.0103	<u>63.37</u>	$t=0.42, w_{idf}=0.60$	0.0476	<u>0.0382</u>	0.0317	0.0138	<u>0.0217</u>	0.0361	9571.94
Genres															
$t=0.30$	0.0030	0.0035	0.0044	0.0007	0.0018	0.0049	4119.41	$t=0.42, w_{idf}=0.60$	0.0376	0.0316	<u>0.0284</u>	0.0105	0.0178	0.0320	1626.74
$t=0.80$	0.0016	0.0022	0.0022	0.0003	0.0013	0.0027	<u>235.48</u>	Pearson Game Tags							
Others															
Details	0.0440	0.0334	0.0257	0.0134	0.0199	0.0310	21246.19	$t=0.55, w_{idf}=0.60$	0.0274	0.0240	0.0210	0.0076	0.0134	0.0235	561.59
Developers	0.0606	0.0554	0.0467	0.0187	0.0343	0.0571	<u>1397.43</u>								
Publishers	0.0570	0.0502	0.0402	0.0168	0.0301	0.0477	3021.03								

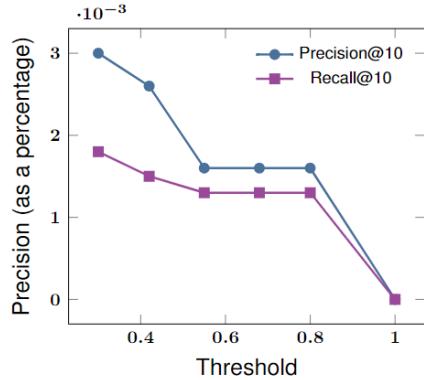
Our IDF hypothesis: is it useful?

Combination	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)
$t=0.30, w_{idf}=0.00$	0.0376	0.0306	0.0262	0.0109	0.0175	0.0302	13594.60
$t=0.30, w_{idf}=0.15$	0.0376	0.0320	0.0269	0.0108	0.0183	0.0304	14949.60
$t=0.30, w_{idf}=0.30$	0.0414	0.0340	0.0290	0.0117	0.0191	0.0328	11879.06
$t=0.30, w_{idf}=0.60$	0.0472	0.0382	0.0318	0.0137	0.0217	0.0364	9379.58
$t=0.42, w_{idf}=0.00$	0.0372	0.0307	0.0264	0.0108	0.0175	0.0303	10241.59
$t=0.42, w_{idf}=0.15$	0.0372	0.0319	0.0271	0.0107	0.0183	0.0307	9034.59
$t=0.42, w_{idf}=0.30$	0.0414	0.0340	0.0289	0.0117	0.0192	0.0328	8600.51
$t=0.42, w_{idf}=0.60$	0.0476	0.0382	0.0317	0.0138	0.0217	0.0361	9571.94
$t=0.55, w_{idf}=0.00$	0.0378	0.0307	0.0262	0.0110	0.0175	0.0302	3052.29
$t=0.55, w_{idf}=0.15$	0.0376	0.0322	0.0265	0.0108	0.0184	0.0302	2954.21
$t=0.55, w_{idf}=0.30$	0.0416	0.0344	0.0278	0.0117	0.0194	0.0313	2873.44
$t=0.55, w_{idf}=0.60$	0.0452	0.0372	0.0306	0.0129	0.0209	0.0348	2313.82
$t=0.68, w_{idf}=0.00$	0.0336	0.0298	0.0245	0.0095	0.0173	0.0281	963.79
$t=0.68, w_{idf}=0.15$	0.0340	0.0297	0.0255	0.0094	0.0169	0.0288	918.73
$t=0.68, w_{idf}=0.30$	0.0366	0.0306	0.0260	0.0100	0.0175	0.0295	865.00
$t=0.68, w_{idf}=0.60$	0.0394	0.0323	0.0248	0.0112	0.0183	0.0286	824.77

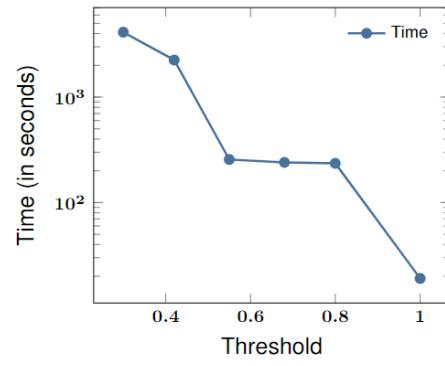
$t=0.80, w_{idf}=0.00$	0.0328	0.0288	0.0244	0.0091	0.0165	0.0276	850.73
$t=0.80, w_{idf}=0.15$	0.0334	0.0291	0.0248	0.0093	0.0164	0.0280	806.09
$t=0.80, w_{idf}=0.30$	0.0354	0.0298	0.0247	0.0096	0.0169	0.0278	752.47
$t=0.80, w_{idf}=0.60$	0.0392	0.0309	0.0235	0.0111	0.0175	0.0271	720.42
$t=1.00, w_{idf}=0.00$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	69.07
$t=1.00, w_{idf}=0.30$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	86.11
$t=1.00, w_{idf}=0.60$	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	68.21
$wt=0.75, w_{idf}=0.00$	0.0376	0.0306	0.0262	0.0109	0.0175	0.0302	23142.23
$wt=0.75, w_{idf}=0.30$	0.0432	0.0364	0.0304	0.0122	0.0205	0.0344	19974.60
$wt=0.75, w_{idf}=0.60$	0.0472	0.0381	0.0317	0.0137	0.0216	0.0363	20120.96
$wt=1.00, w_{idf}=0.00$	0.0360	0.0296	0.0243	0.0104	0.0170	0.0280	8675.47
$wt=1.00, w_{idf}=0.30$	0.0410	0.0334	0.0280	0.0117	0.0191	0.0322	8134.27
$wt=1.00, w_{idf}=0.60$	0.0460	0.0360	0.0286	0.0135	0.0207	0.0329	6528.09

It has proven to be useful across different combinations
 Boosting the scores of uncommon tags =better results.

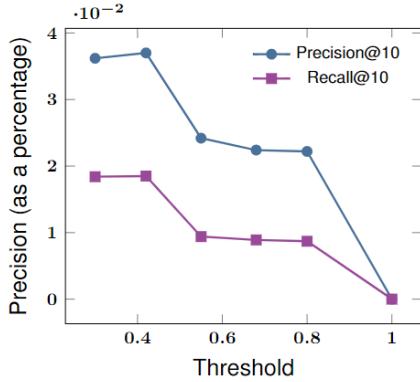
CBF: precision and recall vs time



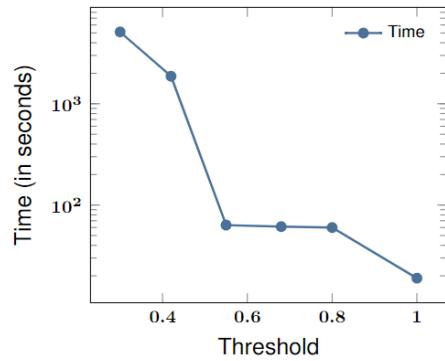
(a) Genres (Precision@5 and Recall@10 vs LSH index threshold)



(b) Genres (Time vs LSH index threshold)

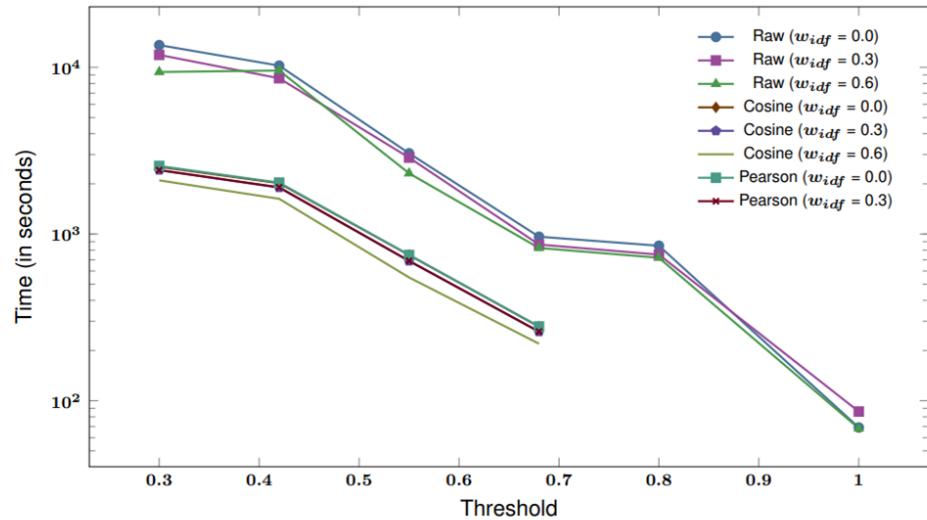
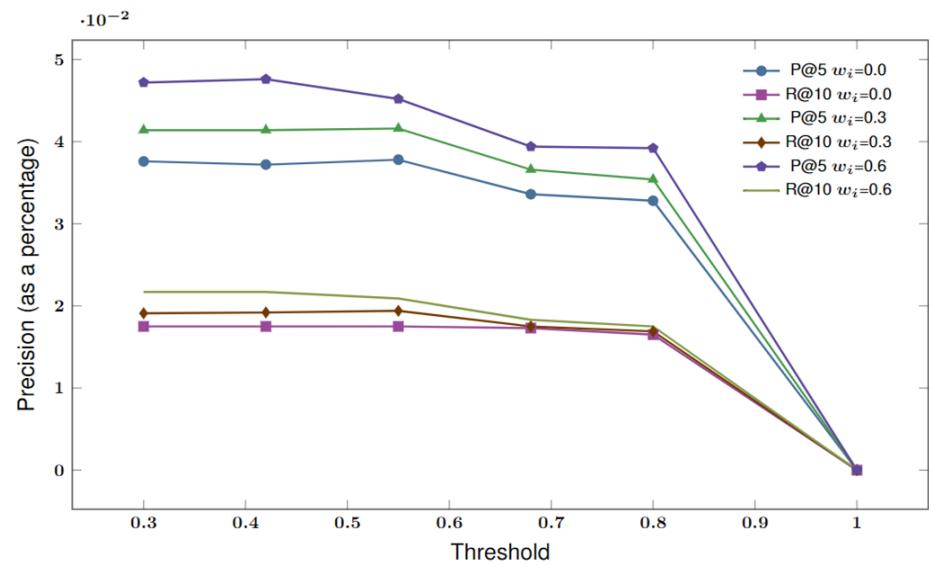


(c) Categories (Precision@5 and Recall@10 vs LSH index threshold)



Precision and recall resemble execution time logarithmically for genres and categories the more increase the threshold (less items to check)

CBF: precision and recall vs time



Same happens for our Game Tags based recommender system
(Cosine and Pearson graphs omitted for brevity)

Our findings: Collaborative Filtering

Results for Collaborative Filtering

Separated by each recommender system

t , the LSH ensemble threshold

t_{rel} means the minimum playtime relative to the top played game for a user to include it in the MinHash

Underscored highlights the best combination for each recommender system

Bold highlights the best overall for Collaborative Filtering

Combination	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)
Raw (Linear)	0.0530	0.0401	0.0314	0.0146	0.0220	0.0348	<u>65261.46</u>
Raw (Log)	0.06⁹²	0.0528	0.0401	0.0191	0.0289	0.0438	65443.07
Raw (Square Root)	0.0594	0.0479	0.0384	0.0164	0.0267	0.0399	65674.90
Cosine (Linear)	0.1400	0.1040	0.0940	0.0490	0.0740	0.0346	61063.86

Combination	P@5	P@10	P@20	R@5	R@10	R@20	Time (s)
$t_{rel}=0.60, t_{lsh}0.60$	0.3100	0.272	0.208	0.0710	0.1225	0.1879	6,420
$t_{rel}=0.60, t_{lsh}0.80$	0.3260	0.286	0.2125	0.0755	0.1289	0.1953	5,160
$t_{rel}=0.80, t_{lsh}0.60$	0.342	0.274	0.2110	0.0798	0.124	0.1896	7,020
$t_{rel}=0.80, t_{lsh}0.80$	0.348	0.277	0.2065	0.0813	0.1256	0.1861	5,760

Table 4.3: Precision and recall results for CF. Parameters used: $t_{rel} = 0.6$, $t = 0.8$. User similarity followed by the normalization approach in parentheses.

Content-Based vs Collaborative

	1	2	3	4	5	6	7
Details	0.0440	0.0334	0.0257	0.0134	0.0199	0.0310	21246.19
Developers	<u>0.0606</u>	<u>0.0554</u>	<u>0.0467</u>	<u>0.0187</u>	<u>0.0343</u>	<u>0.0571</u>	<u>1397.43</u>
Publishers	0.0570	0.0502	0.0402	0.0168	0.0301	0.0477	3021.03
Pearson (Linear)	0.1954	0.1672	0.1281	0.0581	0.0983	0.1512	<u>94042.03</u>
Pearson (Log)	0.2136	0.1753	0.1356	0.0632	0.1030	0.1587	95043.25
Pearson (Square Root)	0.2056	0.1787	0.1389	0.0605	0.1048	0.1625	95983.37

Concluding remarks

Implemented from scratch:

- Data crawler (using SQL)
- Tag scrapper (using Scrapy)
- Playtime normalizer
- Recommender Systems
- Own experiments

We found out CF outperforms CBF, but further optimizations could make everything viable.

Future work

- We would have loved testing more LSH Ensemble thresholds to see if time efficiency is worth lower precision
- More data and comparers, as well as more sophisticated methods (like Artificial Intelligence, specially in our Game Details recommender system)
- We would like to see the performance and precision impact of LSH Ensemble in different domains (music, movies, etc.)
- Real-world deployment and viability
- Further optimization + Using our recommender systems as baselines for performance

The end

Thanks for your attention!



Presentation by Jorge González Gómez. Special thanks to Alejandro Bellogin



To the extent possible under law, JORGE GONZALEZ GOMEZ has waived all copyright and related or neighboring rights to Building a videogame recommendation system from scratch based on user and game data - Presentation. This work is published from: España.

Personal example of discovery queue



More examples of “troll” reviews

Zombie game: troll reviews, or they really like it?
And “how much” do they like it?

152 people found this review helpful
100 people found this review funny  12

 Recommended
453.0 hrs on record

Posted: 22 February, 2022

EARLY ACCESS REVIEW

It has forklifts.

How cool is that?

15 people found this review helpful
2 people found this review funny  1

 Recommended
9.3 hrs on record

Posted: 11 April, 2022

EARLY ACCESS REVIEW

Longer reviews are not
trolls, but do they like
the game more than the
player with 453 hrs?

This game is amazing despite all the negative criticism given to it, not to
is “bad” in it’s current state is kind of wrong. The potential matters the mo

24 people found this review helpful
18 people found this review funny  4

 Recommended
0.6 hrs on record

Posted: 10 January

EARLY ACCESS REVIEW

i wish i had friends

95 people found this review helpful
28 people found this review funny  16

 Recommended
18.3 hrs on record

Posted: 30 June, 2022

EARLY ACCESS REVIEW

cheeseburg 

Caveat: unable to determine if people
with 0.6hrs are more interested than
one with 11.2hrs who actually disliked
the game

90 people found this review helpful
4 people found this review funny  16

 Not Recommended
11.2 hrs on record

Posted: 22 February, 2022

Product received for free

EARLY ACCESS REVIEW

TL;DR: Extremely buggy. Needs another year to stew and polish at
the minimum. I would not spend \$20 on this, let alone the \$30 it is
planned to become. Luckily, I didn't have to.

Different hybrid approaches

Method	Description
Weighted	Each recommender system is assigned a weight, and the final score is the weighted sum of the scores from each recommender system.
Switching	Each recommender system is assigned a threshold, and the final score is the score from the recommender system that passes the threshold.
Mixed	The final score is a combination of the scores from each recommender system.
Feature Combination	The features from each recommender system are combined to create a new recommender system.
Cascade	The first recommender system is used to create a list of recommendations, and then a second recommender system is used to re-rank the list of recommendations.
Feature Augmentation	The output of a recommender system is used as an input for another recommender system.
Meta-level	The model learned by a recommender system is used as an input for another recommender system.