

Санкт–Петербургский государственный университет

*Альберг Анастасия Геннадьевна*

Выпускная квалификационная работа

*Разработка методов интеллектуального анализа  
данных об игровых предпочтениях пользователей  
онлайн-сервиса Steam*

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005 «Прикладная математика,  
фундаментальная информатика и программирование»

Научный руководитель:

доцент, кафедра технологии программирования,  
к.ф.-м.н. Раевская Анастасия Павловна

Рецензент:

профессор, кафедра математического  
моделирования энергетических систем,  
д.ф.-м.н. Крылатов Александр Юрьевич

Санкт-Петербург

2021 г.

# Содержание

<b>Введение</b> . . . . .	3
<b>Постановка задачи</b> . . . . .	4
<b>Обзор литературы</b> . . . . .	5
<b>Глава 1. Основные понятия рекомендательных систем</b> . . . .	8
1.1. Общая постановка задачи . . . . .	8
1.2. Коллаборативная фильтрация . . . . .	10
1.2.1 Коллаборативная фильтрация на основе памяти . . .	11
1.2.2 Коллаборативная фильтрация на основе модели . . .	13
1.3. Подход на основе содержания . . . . .	13
<b>Глава 2. Инструменты по сбору данных</b> . . . . .	15
<b>Глава 3. Метод предоставления рекомендаций</b> . . . . .	18
3.1. Описание алгоритма . . . . .	18
3.2. Проверка работы рекомендательной системы . . . . .	19
3.3. Результат работы для конкретного игрока . . . . .	20
<b>Глава 4. Программная реализация</b> . . . . .	22
4.1. Инструменты . . . . .	22
4.2. Сбор и предобработка данных . . . . .	22
4.2.1 Сбор идентификаторов пользователей . . . . .	22
4.2.2 Сбор игр пользователей . . . . .	23
4.2.3 Предобработка данных . . . . .	23
4.2.4 Сбор дополнительных данных . . . . .	24
4.3. Создание рейтинговой таблицы и прогнозирование . . . .	24
4.4. Вывод результатов . . . . .	24
4.5. Пользовательский интерфейс . . . . .	25
<b>Выводы</b> . . . . .	26
<b>Заключение</b> . . . . .	27
<b>Список литературы</b> . . . . .	28

## Введение

В настоящее время происходит стремительный рост объема цифровой информации, доступной для людей, в любой сфере их жизни. Из-за большого разнообразия выбора пользователи зачастую теряются при подборе интересующего их продукта. Именно поэтому становятся всё более востребованными рекомендательные системы, которые широко используются во многих отраслях, включая онлайн-торговлю, кино и средства массовой информации.

В данной работе рассматривается рекомендательная система по подбору компьютерных игр. Хорошая рекомендательная модель может помочь пользователям быстрее находить видеоигры, которые могут им понравиться. Это не только облегчает задачу по поиску потенциально интересной игры для конкретного игрока, но и помогает дистрибьюторам и разработчикам игр увеличить доходы от своих продаж и повысить уровень удовлетворенности клиентов.

Одной из таких платформ, которая может извлечь большую пользу из системы рекомендаций, является Steam [1] - крупнейший в мире онлайн-сервис цифрового распространения видеоигр с более чем 20 миллионами активных пользователей в день и библиотекой из более чем 30 000 игр.

В данной работе поставлена цель разработки метода предоставления рекомендаций для конкретного пользователя Steam посредством анализа данных игрока. Для этого использовались данные с сайтов Steam и Steam Spy, чтобы узнать, какие игры имеются в библиотеке пользователя, и сколько времени он играл в них. Полученные значения времени игры переводились в рейтинговую систему оценок, после чего производился прогноз для игр, которые потенциально могут понравиться пользователю.

В результате работы алгоритма для конкретного пользователя создавалась рекомендация из десяти игр, имеющих максимальную спрогнозированную оценку, а так же имеющих большее количество положительных отзывов.

## Постановка задачи

Основной целью данной работы является создание программного обеспечения для предоставления пользователям онлайн-платформы Steam релевантных рекомендаций по покупке и использованию игр путем анализа данных игроков.

Для достижения цели были поставлены следующие задачи:

1. Сбор данных о 3000х пользователях Steam с помощью инструмента Steam Web API.
2. Обработка собранных данных об игроках с целью исключения аккаунтов с малым количеством игр и закрытых аккаунтов.
3. Сбор данных об играх с помощью сервиса Steam Spy.
4. Анализ популярности полученных игр по количеству положительных оценок пользователей.
5. Создание рейтинговой таблицы, основываясь на времени, проведенном пользователем в играх.
6. Построение рекомендательной системы коллаборативной фильтрации на основе памяти.

После выполнения всех задач можно обрабатывать данные конкретного игрока и строить рекомендации игр, которые с большой вероятностью ему понравятся. Основа реализуемой системы будет оценивать сходство интересов конкретного пользователя с другими игроками, чтобы предложить высоко оцененные игры пользователей с похожими предпочтениями.

## Обзор литературы

Первый этап активного развития рекомендательных систем начался в 1990-е года, когда Дэвид Голдберг, Дэвид Николс и др. создали систему фильтрации писем Tapestry [2] из-за большого количества входящих спам-писем работникам исследовательского центра Херох в Пало-Альто. Разработчики этой системы первыми использовали термин "коллаборативная фильтрация".

Фундаментальное предположение данного подхода заключается в том, что если пользователи  $X$  и  $Y$  имеют схожее поведение (например, покупают, играют, смотрят, слушают), то они будут действовать в отношении других элементов аналогичным образом [3].

Исследования методов коллаборативной фильтрации были популяризированы благодаря соревнованию Netflix Prize в 2006 году, в ходе которого исследователям необходимо было улучшить работу имеющегося у Netflix алгоритма прогнозирования пользовательских оценок фильмов на 10 %. Авторы статьи [4] сделали обзор на основные подходы, применяемые командами во время проведения соревнования.

Системы коллаборативной фильтрации раннего поколения, такие как GroupLens [5], используют данные об оценках клиентов для вычисления сходства между пользователями или элементами и делают прогнозы в соответствии со значениями сходства. Модели коллаборативной фильтрации, основанные на соседстве, широко используются в коммерческих системах, таких как Amazon(американская платформа электронной коммерции) и Barnes and Noble(американская компания по продажам книг), поскольку они просты в реализации и высокоэффективны [6].

Однако у моделей, основанных на соседстве, существует недостаток: большинство пользователей не предоставляет оценки на какой-либо вид товаров, в результате чего матрица взаимодействия клиентов и элементов получается разреженной, что представляет проблему для вычисления рекомендаций. Для решения подобной проблемы существует и другой тип коллаборативной фильтрации - основанный на модели. При таком подходе рекомендательная система разрабатывается с использованием различных

алгоритмов интеллектуального анализа данных и машинного обучения для прогнозирования оценок пользователей. Используются такие методы, как методы байесовских сетей [7], методы кластерного анализа [8] [9], методы латентно-семантического анализа [10].

Кроме коллаборативной фильтрации применяется иной подход для рекомендательных систем, который основан на содержании. Подобные системы анализируют набор описаний элементов, ранее оцененных пользователем, и создают модель или профиль интересов пользователя на основе характеристик объектов. Другими словами, активному пользователю рекомендуются элементы, аналогичные тем, которые ранее от него получали положительную оценку. Этот подход основан на концепции, согласно которой элементы с похожими атрибутами будут оцениваться одинаково.

Чаще всего источником информации для подобных рекомендательных систем являются текстовые документы. В таком случае применяется векторная модель (Vector Space Model)[11] [12], суть которой состоит в представлении каждого документа коллекции в виде точки в многомерном пространстве. Для полного определения векторной модели необходимо рассчитывать веса термов документа. Наиболее часто используемая схема взвешивания - TF-IDF (Частота термина - Обратная частота документа)[13], основана на эмпирических наблюдениях за текстом. Затем, считая расстояния между полученными точками в векторном пространстве, задача подобиия документов разрешается.

Чтобы избежать некоторых ограничений коллаборативной фильтрации и подхода, основанного на описании, создаются рекомендательные системы гибридного типа, например "content-boosted collaborative filtering"[14]. Так же гибридный подход используется в работе К. Христаку и А. Стафилопаса [15] для построения рекомендательной системы фильмов.

Что касается рекомендательной системы Steam, компания Valve ежегодно совершенствует систему рекомендаций платформы. Однако они никогда не предоставляли никакой информации об алгоритме своей рекомендательной системы, для обычных пользователей система похожа на "черный ящик".

Программист Эрик Джонсон в 2019 году попытался погрузиться в

механизм, лежащий в основе рекомендаций Steam. Он провел два месяца, записывая все игры, в которые он играл, которые просматривал и комментировал, а также время игры и HTML-страницы. В конце исследования Эрик Джонсон пришел к выводу, что система в большей степени полагалась на популярность и новизну игр, чем на интересы пользователя. Кроме того, самым удивительным упущением в системе является отсутствие коллаборативной фильтрации [16].

Исследователи-аналитики неоднократно проводили всесторонний анализ данных игровой сети Steam [17] [18], но не затрагивали вопрос рекомендательных систем.

Авторы статьи [19] для создания рекомендательной системы прогнозирования оценки игры реализовали один из методов тематического моделирования, метод латентного размещения Дирихле, и классификатор игр по жанрам с помощью метода k-ближайших соседей.

В работе [20] авторы описали гибридную рекомендательную систему для платформы Steam, которая учитывает не только рекомендации, основанные на жанрах игр, но и на предпочтениях друзей пользователя, а так же добавили в рассмотрение такие признаки, как цена игры и дата выпуска. Хавьер Перес-Маркос и др. в своем исследовании [21] разработали гибридную систему рекомендаций, учитывающую время, затраченное пользователями в играх.

# Глава 1. Основные понятия рекомендательных систем

*Рекомендательные системы (РС)* - это программные инструменты и методы, предоставляющие рекомендации по предметам, которые могут быть полезны пользователю. Предложения касаются различных процессов принятия решений, таких как покупка товаров, прослушивание музыки или чтение новостей в Интернете. РС в первую очередь ориентированы на людей, у которых нет достаточного личного опыта или компетентности, чтобы оценить огромное число альтернативных элементов, которые может предложить сервис.

РС пытается предсказать, какие продукты или услуги являются наиболее подходящими, основываясь на предпочтениях и ограничениях пользователя. Чтобы выполнить эту задачу, рекомендательная система собирает данные пользовательских предпочтений, которые могут быть явно выражены, например, в виде оценок предметов. Так же некоторые РС могут рассматривать переход на определенную страницу продукта как неявный признак предпочтения товаров.

## 1.1 Общая постановка задачи

Для формального определения задачи построения рекомендательной системы необходимо ввести некоторые обозначения:

$U$  - множество пользователей (users, клиентов) системы;

$I$  - множество предметов (items, товаров, ресурсов) системы;

Обозначим через  $S$  набор возможных оценок для рейтинга предметов (например,  $S = \{1, \dots, 5\}$  или  $S = \{\text{нравится, не нравится}\}$ ). Также предполагается, что любой пользователь  $u \in U$  может сделать не более одной оценки для конкретного элемента  $i \in I$ , эта оценка обозначается  $r_{ui}$ . Собранные воедино оценки всех пользователей образуют матрицу кросс-табуляции (матрицу взаимодействия)  $R = \{r_{ui}\}_{u \in U, i \in I}$ .

Например, в таблице 1 представлены рейтинговые оценки четырех человек (Анны, Ивана, Михаила и Марии) по четырем фильмам (Интерстеллар, Титаник, Король Лев и Игры разума). Пустые ячейки означают, что пользователь не смотрел или не оценивал данный фильм. В предложенном



случае Анна является активным пользователем, для которого необходимо дать рекомендации.

**Таблица 1:** Пример матрицы взаимодействий

	Интерстеллар	Титаник	Король Лев	Игры разума
Анна	4	?	5	5
Иван	4	2	1	4
Михаил	4			4
Мария	2	1	3	5

Вспомогательные обозначения:

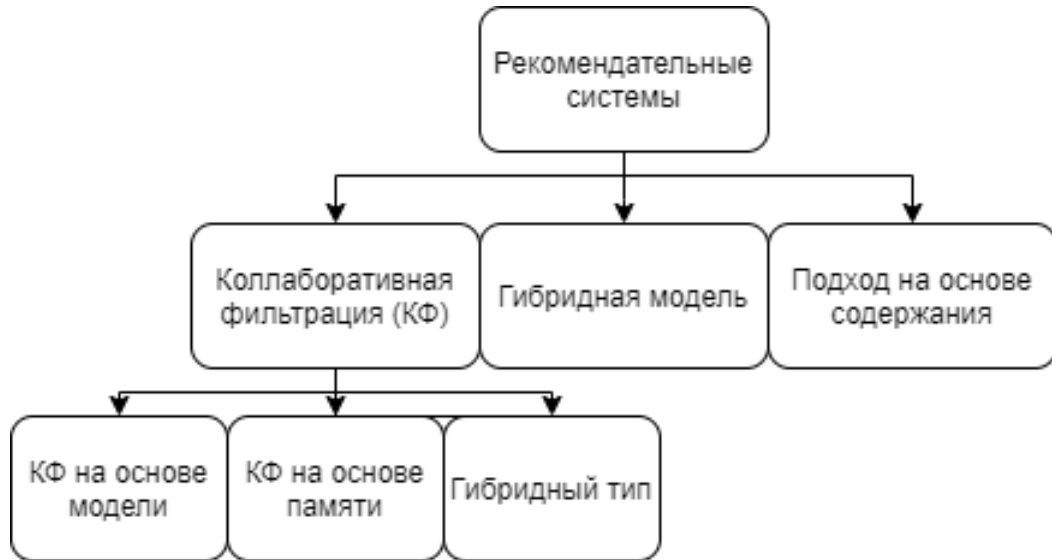
- $U_i$  - подмножество пользователей, оценивших предмет  $i$ ;
- $I_u$  - подмножество предметов, которые оценены пользователем  $u$ ;
- $U_{ij}$  - подмножество пользователей, которые оценили предметы  $i$  и  $j$ , то есть  $U_i \cap U_j$ ;
- $I_{uv}$  - подмножество предметов, которые были оценены двумя пользователями  $u$  и  $v$ , то есть  $I_u \cap I_v$ .

Наиболее распространенные задачи, связанные с рекомендательными системами, - это задачи "best item" и "top-N".

Первая задача состоит в том, чтобы найти для конкретного пользователя  $u$  новый неоцененный элемент  $i \in I \setminus I_u$ , который может заинтересовать  $u$ . Когда оценки доступны, эта задача чаще всего определяется как задача регрессии или классификации, цель которой состоит в том, чтобы изучить функцию  $f : U \times I \rightarrow S$ , которая предсказывает оценку  $f(u, i)$  пользователя  $u$  для нового элемента  $i$ . Затем эта функция используется для рекомендации активному пользователю  $u_a$  элемента  $i$ , для которого оценочная функция имеет наибольшее значение:  $i = \arg \max_{j \in I \setminus I_{u_a}} f(u_a, j)$

Вторая задача состоит в рекомендации активному пользователю  $u_a$  списка  $L(u_a)$ , содержащего  $N$  предметов, которые могут его заинтересовать.

Далее рассмотрим два основных типа рекомендательных систем: коллаборативную фильтрацию, подход на основе содержания. (Рис. 1).



**Рис. 1:** Основные типы рекомендательных систем.

## 1.2 Коллаборативная фильтрация

Подход коллаборативной фильтрации (КФ) опирается на оценки пользователя  $u$ , а также на оценки других клиентов в системе. Ключевая идея заключается в том, что рейтинг клиента  $u$  для неоцененного элемента  $i$ , скорее всего, будет похож на рейтинг предмета  $i$  другого пользователя  $v$ , если  $u$  и  $v$  оценили другие элементы похожим образом. Аналогично, пользователь  $u$ , вероятно, оценит два предмета  $i$  и  $j$  похожим образом, если другие пользователи дали схожие оценки этим двум предметам.

Методы КФ можно разделить на два класса методов, основанных на памяти и основанных на модели.

### 1.2.1 Коллаборативная фильтрация на основе памяти

Алгоритмы КФ на основе памяти используют всю матрицу кросс-табуляции для создания прогноза. Каждый пользователь является частью группы людей со схожими интересами. Находя "соседей" активного пользователя, можно получить для него прогноз предпочтений по неоценным предметам.

Алгоритм КФ на основе памяти вычисляет сходство  $w_{ij}$ , которое отражает расстояние или корреляцию между двумя пользователями или двумя предметами  $i$  и  $j$ . Существует несколько основных методов вычисления сходства между пользователями или предметами:

- Корреляция Пирсона.

Вычисление сходства между двумя пользователями  $u$  и  $v$  производится по формуле:

$$w_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u) (r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}} \quad (1)$$

где суммирование производится по элементам, которые были оценены пользователями  $u$  и  $v$ , а  $\bar{r}_u$  и  $\bar{r}_v$  - средняя оценка совместно оцененных предметов пользователями  $u$  и  $v$  соответственно. Например, из примера в таблице 1  $w_{1,4} = 0,756$ .

Вычисление сходства между двумя предметами  $i$  и  $j$  производится по формуле:

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i) (r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

где суммирование производится по пользователям, которые оценили оба предмета  $i$  и  $j$ , а  $\bar{r}_i$  и  $\bar{r}_j$  - средняя оценка  $i$ -го и  $j$ -го соответственно предмета этими пользователями.

- Косинусная мера близости.

Если  $R$  является матрицей пользовательских оценок размера  $m \times n$ , то сходство между двумя пользователями  $u$  и  $v$  определяется как косинус  $n$ -мерных векторов, соответствующих  $u$ -ой и  $v$ -ой строчкам матрицы  $R$ . Сходство между двумя предметами  $i$  и  $j$  определяется как косинус  $n$ -мерных векторов, соответствующих  $i$ -му и  $j$ -му столбцам матрицы  $R$ .

Векторное косинусное сходство между пользователями  $u$  и  $v$  задается формулой:

$$w_{uv} = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \bullet \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|} = \frac{\sum_{i \in I} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I} r_{ui}^2} \cdot \sqrt{\sum_{i \in I} r_{vi}^2}} \quad (2)$$

Векторное косинусное сходство между элементами  $i$  и  $j$  задается формулой:

$$w_{ij} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|} = \frac{\sum_{u \in U} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U} r_{ui}^2} \cdot \sqrt{\sum_{u \in U} r_{uj}^2}}$$

где “ $\bullet$ ” обозначает скалярное произведение двух векторов.

Получение прогнозов или рекомендаций является наиболее важным этапом в системе коллаборативной фильтрации. В алгоритме КФ, основанном на памяти, на базе полученных значений сходства пользователей или предметов выбирается подмножество ближайших соседей активного пользователя или активного предмета. Затем для генерации прогнозов могут использоваться следующие методы:

- Взвешенная сумма оценок.

Чтобы сделать прогноз для активного пользователя  $a$  по определенному предмету  $i$ , можно использовать средневзвешенное значение всех оценок по этому предмету по формуле:

$$P_{ai} = \bar{r}_a + \frac{\sum_{u \in U_i} (r_{ui} - \bar{r}_u) \cdot w_{au}}{\sum_{u \in U_i} |w_{au}|} \quad (3)$$

где  $\overline{r_a}$  и  $\overline{r_u}$  - среднее значение оценок пользователей  $a$  и  $u$  соответственно по всем предметам за исключением  $i$ , а  $w_{au}$  - вес между пользователями  $a$  и  $u$ . Суммирование проводится по всем пользователям  $u$ , которые оценили предмет  $i$ .

- Среднее арифметическое взвешенное оценок.

Для прогнозирования рейтинга пользователя  $u$  по предмету  $i$  можно использовать среднее арифметическое взвешенное значение по формуле:

$$P_{ui} = \frac{\sum_{n \in I \setminus \{i\}} r_{un} \cdot w_{in}}{\sum_{n \in I \setminus \{i\}} |w_{in}|}$$

где  $w_{in}$  - вес между предметами  $i$  и  $n$ ,  $r_{un}$  - оценка пользователя  $u$  по позиции  $n$ . Суммирование производится по всем предметам за исключением  $i$ .

### 1.2.2 Коллаборативная фильтрация на основе модели

Подход коллаборативной фильтрации, основанный на модели, предполагает использование информации о взаимодействиях пользователя и предметов, а также разработку скрытой модели поведения, которая должна объяснить эти взаимодействия.

В качестве средств для создания таких моделей используют методы машинного обучения. Если оценки пользователей являются категориальными, используются алгоритмы классификации, а для численных оценок могут быть применены регрессионные модели и методы SVD.

## 1.3 Подход на основе содержания

Рекомендательные системы, основанные на контенте (содержании), анализируют набор документов или описаний предметов, которые ранее были оценены пользователем, после чего строят модель или профиль интересов пользователя на основе особенностей предметов.

Большинство рекомендательных систем на основе содержания используют относительно простые модели, например, модель векторного пространства (Vector Space Model, VSM) с базовым взвешиванием TF-IDF. VSM - это представление текстовых документов в векторном пространстве.

Представление документов в VSM делится на два этапа: взвешивание термов и измерение сходства вектора признаков. Наиболее часто используемая схема взвешивания термов - это взвешивание TF-IDF (Частота терма - Обратная частота документа).

$$\text{tf-idf}(t,d,D) = \text{tf}(t,d) \times \text{idf}(t,D)$$

$$\text{tf}(t,d) = \frac{n_t}{\sum_k n_k}$$

где  $n_t$  — число вхождений слова  $t$  в документ  $d$ , а в знаменателе — общее число слов в данном документе.

$$\text{idf}(t,D) = \log \frac{|D|}{|\{d_i \in D | t \in d_i\}|}$$

где в числитель — это число документов в коллекции, а знаменатель — число документов из коллекции  $D$ , в которых встречается слово  $t$

## Глава 2. Инструменты по сбору данных

API (Application Programming Interface) - интерфейс прикладного программирования, который описывает взаимодействие между несколькими программными приложениями.

Компания Valve предоставляет доступ к Steam Web API [22], чтобы разработчики могли использовать данные с платформы Steam для создания собственных приложений. Для его использования требуется наличие ключа Steam Web API.

Данный сервис предоставляет большое количество методов и интерфейсов для получения различной информации, но нас интересует интерфейс ISteamUser с методом GetFriendList, а так же интерфейс IPlayerService с методом GetOwnedGames.

Метод GetFriendList возвращает список друзей пользователя со следующей информацией: "steamid" - идентификатор друга в Steam, "relationship" - фильтр отношений, "friend\_since" - unix время, когда пользователи подружились (Рис. 2), но только в том случае, если профиль находится в открытом доступе.

```
"friendslist": {  
  "friends": [  
    {  
      "steamid": "76561198047134912",  
      "relationship": "friend",  
      "friend_since": 1406190216  
    },  
    {  
      "steamid": "76561198212159657",  
      "relationship": "friend",  
      "friend_since": 1545417513  
    }  
  ]  
}
```

**Рис. 2:** JSON ответ от сервера на запрос о друзьях игрока

Запрос на сервер выглядит следующим образом:

`http://api.steampowered.com/ISteamUser/GetFriendList/v0001/?  
key=KEY&steamid=ID&relationship=friend&format=json`

где KEY - ключ Steam Web API, ID - идентификатор пользователя Steam.

Метод `GetOwnedGames` возвращает количество игр в библиотеке у пользователя ("game\_count") и список игр пользователя со следующей информацией: "appid" - идентификатор игры в Steam, "playtime\_forever" - время в минутах, проведенное пользователем в игре, "playtime\_windows\_forever", "playtime\_mac\_forever", "playtime\_linux\_forever" - время в минутах, проведенное пользователем в игре в соответствующей операционной системе (Рис. 3). Доступ к данным есть только в том случае, если игрок не скрыл свои игры в настройках доступа.

Запрос на сервер выглядит следующим образом:

```
http://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/?  
key=KEY&steamid=ID&format=json
```

где KEY - ключ Steam Web API, ID - идентификатор пользователя Steam.

```
"response": {  
  "game_count": 36,  
  "games": [  
    {  
      "appid": 4570,  
      "playtime_forever": 396,  
      "playtime_windows_forever": 0,  
      "playtime_mac_forever": 0,  
      "playtime_linux_forever": 0  
    },  
    {  
      "appid": 9310,  
      "playtime_forever": 859,  
      "playtime_windows_forever": 0,  
      "playtime_mac_forever": 0,  
      "playtime_linux_forever": 0  
    },  
  ],  
}
```

**Рис. 3:** JSON ответ от сервера на запрос об играх пользователя

Так же кроме сервиса Steam Web API в работе используется API сайта Steam Spy [23], где собраны оценки объёмов продаж предлагаемых в магазине Steam продуктов, а так же разные статистики по играм.



С помощью запроса: <http://steamspy.com/api.php?request=appdetails&appid=ID>, где ID - идентификатор игры Steam, можно получить следующие данные:

- "appid" - идентификатор игры на сайте Steam;
- "name" - название игры;
- "developer" - разработчики игры;
- "publisher" - издатели игры;
- "positive" , "negative" количество положительных и отрицательных оценок игры;
- "ownres" - количество владельцев;
- "average\_forever" , "average\_2weeks" - среднее время игры (наблюдения велись с с марта 2009 года) и за последние две недели;
- "median\_forever" , "median\_2weeks" - медианное значение времени игры (наблюдения велись с с марта 2009 года) и за последние две недели;
- "price" - текущая цена игры в центах;
- "discount" - текущая скидка на игры в процентах.

Для данной работе используются атрибуты: идентификатор игры, название игры, количество положительных и отрицательных оценок игры, среднее время, проведенное в игре, с марта 2009 года.

## Глава 3. Метод предоставления рекомендаций

Пусть нам дан список пользователей, для каждого из них найден список игр, в которые пользователь играл, а так же известно количество часов, проведенное за каждой игрой. На основе информации найдено множество уникальных игр и среднее значение времени игры с марта 2009 года.

### 3.1 Описание алгоритма

На платформе Steam оценка игр просходит с помощью ранжирования "нравится" / "не нравится" , но малое количество пользователей использует этот функционал, поэтому необходимо построить оценочную систему по-другому.

Для построения матрицы взаимодействия (кросс-табуляции) предлагается использовать в качестве показателя оценки пользователя время, проведенное за игрой.

Введем обозначения:

- $t_{ui}$  - время, проведенное пользователем  $u$  за игрой  $i$ ;
- $r_{ui}$  - значение оценки пользователя  $u$  игры  $i$ ;
- $\tau_i$  - среднее значение времени, проведенное за игрой  $i$ , с марта 2009 года.

$$r_{ui} = \begin{cases} 0, & \text{если } t_{ui} = 0 \\ 1, & \text{если } 0 < t_{ui} < 0.4 * \tau_{ui} \\ 2, & \text{если } 0.4 * \tau_{ui} < t_{ui} \leq 0.8 * \tau_{ui} \\ 3, & \text{если } 0.8 * \tau_{ui} < t_{ui} \leq 1.2 * \tau_{ui} \\ 4, & \text{если } 1.2 * \tau_{ui} < t_{ui} \leq 1.6 * \tau_{ui} \\ 5, & \text{если } t_{ui} \geq 1.6 * \tau_{ui} \end{cases} \quad (4)$$

Пример таблицы кросс-табуляции по имеющимся тестовым данным показан на рис. 4

	360450	688130	65540	262150	458760	10	...	753640	262120	458730	360430	393200	65530	98300
76561198047134912	0	0	5	0	0	0	...	0	0	0	0	0	0	0
76561198983924434	0	0	0	0	0	0	...	0	0	0	0	0	0	0
76561197976267165	0	0	5	0	0	0	...	0	0	0	0	0	0	0
76561198007881103	0	0	0	0	0	0	...	0	0	0	0	0	0	0
76561198025227412	0	0	0	0	0	1	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
76561198047698967	0	0	0	0	0	0	...	0	0	0	0	0	0	0
76561198068110550	0	0	0	0	0	0	...	0	0	0	0	0	0	0
76561198139652282	0	0	0	0	0	0	...	0	0	0	0	0	0	0
76561198154070711	0	0	0	0	0	0	...	0	0	0	5	0	0	0
76561198350455459	0	0	0	0	0	0	...	0	0	0	0	0	0	0

**Рис. 4:** Фрагмент матрицы оценок игр

После того, как таблица рейтингов будет получена, запускается алгоритм, основанный на памяти, или другими словами на соседстве, который был подробно описан в пункте 1.2.1 данной работы. Алгоритм состоит из двух основных этапов:

- Вычисление сходства игроков по формуле 1 (корреляция Пирсона) или по формуле 2 (косинусовое сходство).
- Прогнозирование оценки игры  $i$  для конкретного пользователя  $u$  по формуле 3

Таким образом, мы можем восстановить любой неизвестный элемент в таблице оценок.

В дополнении к алгоритму коллаборативной фильтрации на основе памяти создается рейтинг игр, который основывается на количестве положительных отзывов. Показателем для такого ранжирования игр будет величина:  $value = \frac{n_{pos}}{n_{pos} + n_{neg}}$ , где  $n_{pos}$  - число положительных рекомендаций,  $n_{neg}$  - число отрицательных рекомендаций.

Далее рассмотрим результаты работы рекомендательной системы и оценку ее прогнозов.

## 3.2 Проверка работы рекомендательной системы

Для того, чтобы оценить работу построенной системы, случайным образом бралась тестовая выборка из 100 игроков, около 10 % от общего числа рассматриваемых пользователей. Для каждого из них находился вектор, состоящий из вычисленных значений схожести с другими игроками.

Затем для каждого известного значения оценки рассматриваемого игрока (то есть для тех игр, которые уже есть у него в библиотеке) прогнозировалась оценка для этого элемента.

Если обозначить  $p_{ui}$  за спрогнозируемую оценку игры  $i$  для игрока  $u$ , а  $r_{ui}$  за известную рейтинговую оценку игры  $i$  для игрока  $u$ , то можно посчитать среднеквадратическую ошибку модели по формуле:

$$RMSE = \sqrt{\frac{1}{n} \sum_{(u,i)} (p_{ui} - r_{ui})^2}$$

Для полученной рекомендательной системы имеем оценки, представленные на рис. 5 и рис. 6

Средняя RMSE с метрикой cosine 1.3589571355328132

**Рис. 5:** RMSE для косинусовой меры близости

Средняя RMSE с метрикой pearson 1.1254247525933636

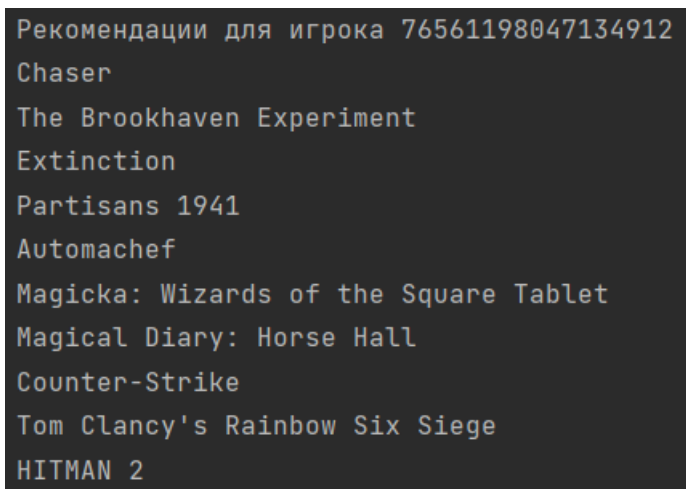
**Рис. 6:** RMSE для корреляции Пирсона

### 3.3 Результат работы для конкретного игрока

Для предоставления рекомендаций конкретному игроку, алгоритм начинает свою работу по прогнозированию точно так же, как описано в пункте 3.1., в результате получем вектор спрогнозированных оценок для каждой игры, в которую пользователь не играл. После чего идет поиск игр с максимальным значением спрогнозированной оценки.

Для предоставления конечного списка рекомендаций случайным образом выбирается семь игр с максимальным рейтингом из результата работы системы коллаборативной фильтрации, а так же три дополнительные игры, основываясь на числе положительных рекомендаций пользователей.

На рис. 7 представлен результат работы алгоритма для пользователя с идентификатором 76561198047134912.



```
Рекомендации для игрока 76561198047134912
Chaser
The Brookhaven Experiment
Extinction
Partisans 1941
Automachef
Magicka: Wizards of the Square Tablet
Magical Diary: Horse Hall
Counter-Strike
Tom Clancy's Rainbow Six Siege
HITMAN 2
```

**Рис. 7:** Рекомендации для пользователя с id 76561198047134912

## Глава 4. Программная реализация

С кодом данной рекомендательной системы можно ознакомиться на странице репозитория GitHub [24].

### 4.1 Инструменты

В качестве языка программирования был выбран высокоуровневый язык программирования Python, как популярный инструмент разработки с большим выбором библиотек, помогающих в работе над проектом.

В процессе реализации рекомендательной системы были использованы следующие библиотеки Python:

- requests - библиотека, позволяющая отправлять HTTP запросы;
- json - библиотека, позволяющая кодировать и декодировать данные в удобном формате;
- pandas — библиотека для обработки и анализа данных;
- numpy — математическая библиотека с поддержкой многомерных массивов;
- statistic — библиотека, предоставляющая функции для расчета математических статистик числовых данных;
- tkinter - библиотека для разработки графического интерфейса.

### 4.2 Сбор и предобработка данных

Весь процесс сбора и предобработки данных делится на четыре этапа. Далее кратко приведем описание последовательности выполненных действий.

#### 4.2.1 Сбор идентификаторов пользователей

Для того, чтобы собрать id пользователей Steam, было решено начать взаимодействие с данными игрока с идентификатором 76561198047134912.

Для этого с помощью библиотеки request был отправлен get-запрос к Steam Web API на получение данных о друзьях данного пользователя, после чего был получен ответ со списком друзей в формате JSON.

Далее эта процедура повторялась с друзьями исходного игрока, а затем и с их друзьями и так далее. Данный алгоритм проверяет наличие пустых ответов на запросы в случае закрытых профилей. И заканчивает сбор данных, когда количество запросов достигает 3000.

В результате этой части программы имеем список с идентификаторами пользователей Steam размера  $1 \times 2990$ .

#### **4.2.2 Сбор игр пользователей**

На этом шаге для каждого пользователя, идентификатор которого известен из предыдущего шага, делается соответствующий запрос к Steam Web API на получение списка игр, находящихся у игрока в библиотеке. При этом обрабатывается случай, когда пользователь закрыл доступ к просмотру его игр или доступ к ним есть только у его друзей. В результате в рассмотрении остался список с 1101 пользователем и список игр этих игроков.

#### **4.2.3 Предобработка данных**

В процессе обработки полученных на предыдущих этапах данных было поставлено несколько задач.

Во-первых, необходимо убрать из рассмотрения игры и дополнения к ним, которые пользователь ни разу не запускал, то есть у которых время игры равнялось нулю.

Во-вторых, нужно удалить из базы данных игроков, у которых в библиотеке находится меньше пяти игр, так как подавляющее большинство пользователей создает такие аккаунты с целью обхода правил сервиса, которые не позволяют играть после блокировки аккаунта за нарушение пользовательского соглашения.

Окончательно получаем 926 игроков, на которых будет обучаться рекомендательная система, а так же список из 9689 уникальных игр, в кото-

рые играли изучаемые игроки.

#### 4.2.4 Сбор дополнительных данных

Далее путем запроса к API сайта Steam Spy находим для каждой уникальной игры среднее время, которое в ней проводили пользователи Steam. Дополнительно запрашиваем данные о положительных и отрицательных рекомендациях пользователей. Вычисляем относительное число положительных оценок для каждой игры и сортируем массив по убыванию этого параметра.

### 4.3 Создание рейтинговой таблицы и прогнозирование

Изначально создается таблица, заполненная нулями, размера  $926 \times 9689$ , после чего согласно формуле 4 высчитываются рейтинговые оценки игр.

Были реализовали две меры сходства: корреляция Пирсона (формула 1) и косинусное сходство (формула 2), а так же была создана функция, которая делает прогноз оценки игры  $i$  пользователем  $u$  по формуле 3.

#### 4.4 Вывод результатов

Для оценки работы алгоритма с помощью пакета random случайным образом из выборки отбиралось 100 игроков. Для каждой игры тестовых игроков, оценка которой известна, делался прогноз и считалась ошибка RMSE. Этот процесс был проведен для каждой из метрик близости и после этого посчитано среднее значение ошибки на тестовой выборке.

В результате имеем, что значение ошибки при выборе корреляции Пирсона в качестве меры близости оказалось ниже, чем для косинусового сходства. Поэтому для рекомендаций конкретному пользователю решено использовать корреляцию Пирсона для вычисления сходства между игроками.

На финальном этапе создания списка рекомендованных игр мы имеем массив ранее не оцененных игр со спрогнозированными оценками и массив



игр, упорядоченных по убыванию относительного числа положительных рекомендаций других пользователей. В итоге с помощью пакета random выбирается 7 игр с максимальным предсказанным рейтингом и 3 игры из списка игр с большим количеством положительных оценок.

## 4.5 Пользовательский интерфейс

Для предоставления рекомендаций конкретному пользователю был реализован графический интерфейс при помощи библиотеки tkinter. Для взаимодействия с пользователем используются виджеты Entry и Button. Пользователю предлагается ввести идентификатор игрока Steam в поле виджета Entry, который используется для приема однострочных текстовых данных.

После нажатия на кнопку «Дать рекомендацию» происходит проверка введенного идентификатора. Если данного id не существует или найденный аккаунт является закрытым, соответствующая информация выводится с помощью виджета Label. Если же найденный аккаунт публичный, то запускается рекомендательная система, описанная в главе 3. Результаты работы алгоритма представляются в виде списка из десяти виджетов Label.

На рисунке 8 представлен графический интерфейс с рекомендациями для игроков с идентификаторами 76561198047134912 и 76561198047521627.

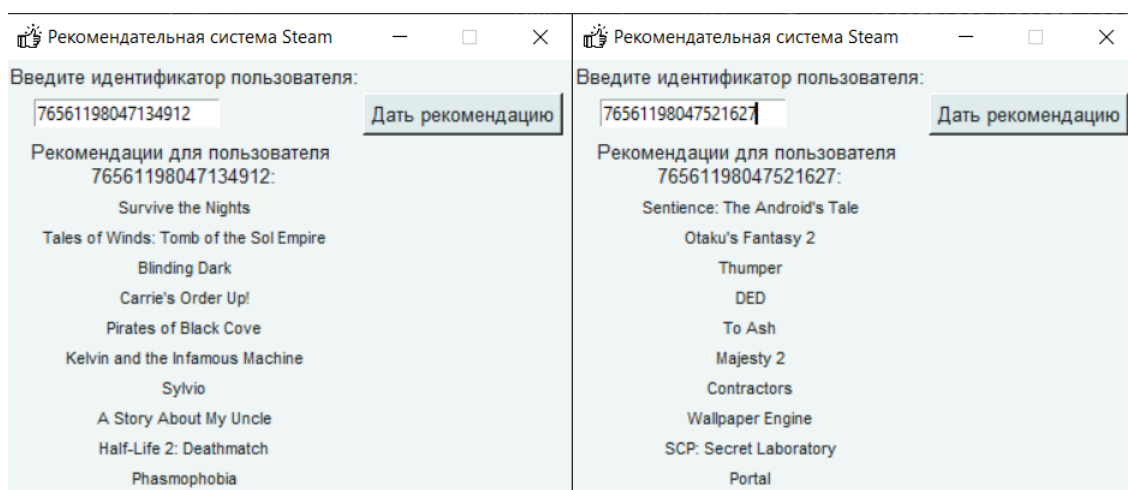


Рис. 8: Результаты работы для двух пользователей

## Выводы

Таким образом, анализируя данные пользователя из онлайн-платформы дистрибуции игр Steam, мы можем получить необходимую информацию о наиболее предпочитаемых играх каждого конкретного пользователя. И на основе этих данных можем сформировать список игр, которые могли бы понравиться пользователю.

В связи с большим количеством продуктов, предлагаемых на платформе Steam, данная рекомендательная система позволит игрокам быстрее находить игры, подходящие им по предпочтениям.

Реализованная система дает рекомендации с небольшой погрешностью. Но при этом можно уменьшить погрешность путем добавления в рассматриваемую систему иного подхода, например кооперативную фильтрацию на основе модели или подхода на основе содержания. Это позволит избавиться от недостатка КФ на основе памяти, связанного с наличием разреженной матрицы большого размера.

## Заключение

В результате разработки метода интеллектуального анализа данных игроков онлайн-сервиса дистрибуции игр Steam была создана рекомендательная система игр.

В рамках данной работы было собрано большое количество различных данных с помощью интерфейса API сайтов Steam и Steam Spy. После чего проводилась обработка данных с помощью популярных библиотек языка Python.

На основе полученных данных была разработана рекомендательная система коллаборативной фильтрации на основе памяти, основываясь на времени, проведенном пользователями в играх. Так же был проведен анализ игр по количеству положительных отзывов.

В результате комбинации результатов этих двух моделей, для любого пользователя Steam создается рекомендация из десяти игр.

## Список литературы

- [1] Официальный сайт магазина Steam. <https://store.steampowered.com/>
- [2] Goldberg D., Nichols D., Oki B. M., Terry D. «Using collaborative filtering to weave an information Tapestry»// Communications of the ACM, 1992, Vol. 35, No. 12, P. 61–70.
- [3] Goldberg K., Roeder T., Gupta D., Perkins C. «Eigentaste: a constant time collaborative filtering algorithm»// Information Retrieval, 2001, Vol. 4, No. 2, P. 133–151.
- [4] Feuerverger A., He Y., Khatri S. «Statistical significance of the Netflix challenge»// Statistical Science, 2012, Vol. 27, No. 2, P. 202–231.
- [5] Resnick P.,Iacovou N.,Suchak M.,Bergstrom P.,Riedl J. «Grouplens: an open architecture for collaborative filtering of netnews»// Computer-Supported Cooperative Work, 1994, P. 175–186.
- [6] Linden G., Smith B., York J. «Amazon.com recommendations: item-to-item collaborative filtering»//IEEE Internet Computing, 2003, Vol. 7, No. 1, P. 76–80.
- [7] Miyahara K., Pazzani M. J. «Collaborative filtering with the simple Bayesian classifier»// Pacific Rim International Conference on Artificial Intelligence, 2000, P. 679–689.
- [8] Gong S. «A collaborative filtering recommendation algorithm based on user clustering and item clustering»// Journal of Software, 2010, Vol. 5, No. 7, P. 745–752.
- [9] Ungar L. H.,Foster D. P. «Clustering methods for collaborative filtering»// AAAI Technical Report, 1998, P. 114–129.
- [10] Hofmann T. «Latent semantic models for collaborative filtering»// ACM Transactions on Information Systems, 2004, Vol. 22, No. 1, P. 89–115.

- [11] Lee D.L., Chuang H., Seamons K. «Document ranking and the vector-space model»// IEEE Software, 1997, Vol. 14, No. 2, P. 67-75.
- [12] Salton G., Wong A., Yang C. S. «A vector space model for automatic indexing»// Communications of the ACM, 1975, Vol. 18, No. 11, P. 613–620.
- [13] Salton G., Buckley C. «Term-weighting approaches in automatic text retrieval»// Information Processing & Management, 1988, Vol. 24, No. 5, P. 513-523.
- [14] Melville P. , Mooney R. J., Nagarajan R. «Content-boosted collaborative filtering for improved recommendations»// AAAI, 2002, P. 187–192.
- [15] Christakou, C., Stafylopatis, A. «A hybrid movie recommender system based on neural networks»// International Conference on Intelligent Systems Design and Applications, 2005, P. 500–505.
- [16] Johnson E. «A deep dive into steam’s discovery queue»  
[https://www.gamasutra.com/blogs/ErikJohnson/20190404/340061/A\\_Deep\\_Dive\\_Into\\_Steams\\_Discovery\\_Queue.php](https://www.gamasutra.com/blogs/ErikJohnson/20190404/340061/A_Deep_Dive_Into_Steams_Discovery_Queue.php).
- [17] Sifa R., Drachen A., Bauckhage C. «Large-scale cross-game player behavior analysis on Steam»// AIIDE, 2015, P. 198-204.
- [18] O’Neill J. W. M., Vaziripour E., Zappala D. «Condensing steam: Distilling the diversity of gamer behavior» <https://steam.internet.byu.edu>.
- [19] Ahmad Kamal A. S .b., Saaidin S., Kassim M. «Recommender system: Rating predictions of Steam games based on genre and topic modelling»//IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), 2020, P. 212-218.
- [20] Gong J., Ye Y., Stefanidis K. «A hybrid recommender system for Steam games»// Communications in Computer and Information Science, 2020, Vol. 1197, P. 133-144.
- [21] Perez-Marcos J., Martin-Gomez L., Jimenez-Bravo D. M., Lopez V. F., Moreno-Garcia M.N. «Hybrid system for video game recommendation based

on implicit ratings and social networks»// Journal of Ambient Intelligence and Humanized Computing, 2020, Vol. 11, No. 11, P. 4525–4535.

- [22] Официальный сайт документации Steam Web API.  
[https://developer.valvesoftware.com/wiki/Steam\\_Web\\_API/](https://developer.valvesoftware.com/wiki/Steam_Web_API/)
- [23] Официальный сайт сервиса Steam Spy. <https://steamspy.com/>
- [24] Программная реализация рекомендательной системы на GitHub.  
[https://github.com/albergnaa/vkr\\_steam\\_recommender\\_system](https://github.com/albergnaa/vkr_steam_recommender_system)