Автономная некоммерческая организация высшего образования  
"Университет Иннополис"

**Система рекомендаций по фильмам,   
основанная на исходящих ссылках из Википедии**

**отчет по программе дополнительного профессионального образования**

**«Data Science»**

Выполнила:

Суслова Ольга Николаевна

2021

**Содержание**

[Введение - 3 -](#_Toc90693845)

[Часть 1. Существующие методы и подходы к разработке рекомендательных систем - 4 -](#_Toc90693846)

[Часть 2. Анализ существующих источников данных - 6 -](#_Toc90693847)

[Часть 3. Анализ и обработка используемых данных - 8 -](#_Toc90693848)

[Часть 4. Построение модели выбора ближайших фильмов - 14 -](#_Toc90693849)

[Часть 5. Построение модели рекомендаций по интересам пользователя или набору выбранных фильмов - 17 -](#_Toc90693850)

[Часть 6. Результаты применения моделей и методов - 18 -](#_Toc90693851)

[Заключение - 20 -](#_Toc90693852)

[Библиографический список - 21 -](#_Toc90693853)

[Приложение 1. Датасеты для построения рекомендательной системы по фильмам - 22 -](#_Toc90693854)

# Введение

Сегодня, во времена научно-технического прогресса, все больше и больше людей в мире используют интернет. В то же самое время у компаний растут объемы данных о пользователях, которые им надо как-то использовать, чтобы получить с этого некоторую выгоду. Для решения данной задачи развиваются многочисленные математические и интеллектуальные методы обработки и анализа данных. Среди них предсказание вероятности оттока пользователя и его удержание, оценка стоимости жилья, кредитный скоринг, компьютерное зрение, обработка текста и многие другие задачи.

Одной из таких задач является разработка рекомендательных систем, позволяющих понимать и предсказывать, какой товар лучше предложить пользователю, если он интересуется некоторыми другими товарами.

Однако рекомендательные системы могут быть применены и в других сферах. Например, в медицине, образовании, производстве, менеджменте и многих других, где они могут быть использованы для предварительного анализа входных данных по той или иной проблеме и выводе информации по возможным вариантам ее решения, что упростит пользователю выбор окончательного способа решения.

В настоящее время разработано множество рекомендательных моделей и систем, основанных на разных признаках и использующих различные подходы. Однако их всегда можно дорабатывать и улучшать, искать новые методы и варианты разработки моделей.

В рамках данной работы в соответствии с заданием будет построена рекомендательная система для фильмов, основанная на исходящих ссылках в описании фильмов на Википедии.

# Часть 1. Существующие методы и подходы к разработке рекомендательных систем

Рекомендательная система – персонализированная технология фильтрации информации, используемая для прогнозирования, понравится ли конкретному пользователю определенный элемент (задача прогнозирования) или для идентификации набора из N элементов, интересных определенному пользователю (задача рекомендации топ-N). [5]

Существуют два основных класса для построения рекомендательных систем: персонализированные и неперсонализированные. Неперсонализированные рекомендации строятся без учета личных предпочтений пользователей, когда система сортирует популярные товары по средней оценке рейтинга или другим важным критериям товаров, важно, что при этом не учитываются данные о конкретных пользователях. Персонализированные рекомендации для построения используют всю доступную информацию о пользователе, его историю активности и предпочтения для того, чтобы предоставить ему то, что будет ему наиболее интересно. [2]

Поскольку неперсонализированные рекомендации не учитывают имеющиеся знания о конкретных пользователях, они обычно менее точны по сравнению с персонализированными методами. Поэтому на практике обычно используют персонализированные методы рекомендаций.

На данный момент существуют два основных подхода к построению персонализированных рекомендательных систем – контент-ориентированная и коллаборативная фильтрация. [1]

Суть контент-ориентированного подхода заключается в том, что пользователи сопоставляются с имеющимся контентом о товарах, которые им нравились или были ими куплены. Здесь в первую очередь важны атрибуты пользователей и товаров. Например, для рекомендаций к фильмам используются такие признаки, как режиссер, актеры, продолжительность фильма, жанр и т.д., чтобы найти сходство между фильмами. Целью контент-ориентированных методов является создание «профиля» для каждого пользователя и каждого товара.

При этом к недостаткам данного подхода можно отнести сложность рекомендаций для нового пользователя, о котором на данный момент времени нет никакой информации; формирование четких ограниченных групп похожих товаров и услуг, что мешает делать рекомендации из других товаров и услуг (хотя это легко может быть преодолено некоторыми случайными рекомендациями их других групп, но это уже не относится к данному методу); низкое качество рекомендаций при малом количестве информации о товарах.

Основополагающее предположение подхода коллаборативной фильтрации заключается в том, что если два человека покупают аналогичные товары, то второй, скорее всего, купит товар, который купил первый, чем товар, который купил случайный человек. В отличие от контентно-ориентированного подхода, здесь нет признаков, соответствующих пользователям или предметам. Для определения интересности того или иного товара пользователю используется т.н. матрица полезности.

Существует две широких категории, на которые можно разделить коллаборативную фильтрацию – основанные на памяти и основанные на модели. В первом случае запоминается матрица полезности товаров для пользователей, и рекомендации составляются путем запроса данного пользователя к остальной части матрицы полезности. Во втором - создаются представления пользователей и предметов из матрицы полезности. В методах, основанных на памяти, результаты очень легко получать и интерпретировать, но как только данные становятся слишком разреженными, производительность ухудшается. В случае использования методов на основе модели, к недостаткам можно отнести снижение интерпретируемости, поскольку неизвестно, что именно означают элементы векторов пользователей/товаров.

Кроме того, для построения рекомендательной системы без учителя можно использовать кластеризацию. Данный метод используется в случаях, когда мало данных. При этом имеющиеся наблюдения кластеризуются на основе имеющихся признаков, а затем на основании размеченных кластеров назначаются соответствующие рекомендации. Данный метод используется в случаях, пока не будет получено достаточно данных. Так же кластеризация может быть использована для создания мета-признаков для объектов.

Так же существуют такие подходы, как демографический (рекомендация на основе демографической группы пользователя), utility-based (разработка функции полезности utility function для конкретного пользователя), knowledge-based (рекомендация, основанная на базе знаний о соотношениях интересов пользователя и существующих товаров; далее по этой базе знаний оценивается полезность для конкретного пользователя); и прочие. [4]

Гибридные методы объединяют существующие методы и заключаются в некотором смешении данных подходов. Например, из одной модели берется некоторый результат и подается на вход другой. Или в одних условиях работает одна модель, а в других – другая, и так далее.

В данной работе будет рассмотрен следующий подход к построению рекомендательных систем. Будет построена рекомендательная система, основанная на текстовом описании данных. В соответствии с заданием, будет разработана рекомендательная система фильмов, основанная исключительно на исходящих ссылках статей Википедии, посвященных данным фильмам. Далее с помощью классификатора будет реализована рекомендация на основании интересов пользователя. По сути это рекомендательная система, основанная на содержании. Персонализированная, когда мы применяем ее к конкретным интересам пользователя.

# Часть 2. Анализ существующих источников данных

В рамках данного задания данные были предоставлены в самом задании [6] и содержат 10000 записей о существующих фильмах базы Википедии со следующими полями (после переименования числовых названий полей):

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 10000 non-null object

1 Description 10000 non-null object

2 Links 10000 non-null object

3 Rating1 5584 non-null object

4 Rating2 5584 non-null object

dtypes: object(5)

memory usage: 390.8+ KB

В предоставленном датасете мы имеем название фильма, его описание, исходящие ссылки на статьи Wikipedia и две оценки соответствующего фильма. Более подробный анализ данных будет проведен в следующей части.

Данные были уже предоставлены, однако так же были проанализированы возможные альтернативные источники данных для построения данной модели. Полный их список приведен в приложении.

В первую очередь рассмотрим ресурс

* <https://dumps.wikimedia.org> [1]:
  + <https://dumps.wikimedia.org/enwiki> [1.a]
  + <https://dumps.wikimedia.org/ruwiki> [1.b]

Это основной источник по выгрузке подобных данных из Википедии по медиаконтенту. Содержит информацию о существующих на Wiki статьях, и прочие данные, как на русском, так и на английском языках.

Один из вариантов парсинга данных с wikimedia.org описан в книге [3]. Другой – в источнике на Гитхабе [wikipedia-film-database](https://github.com/piki/wikipedia-film-database) [2] c полученным датасетом, расположенном по адресу <https://oracleofbacon.org/data.txt.bz2> [2.a].

Различные варианты парсинга датасетов, сами датасеты и даже уже построенных рекомендательных систем можно найти на Гитхабе: <https://github.com/topics/movies-dataset> [3].

Например, интересны следующие работы:

* [The-Movie-Cinema](https://github.com/kishan0725/The-Movie-Cinema) [3.a]
* [Content-Based-Movie-Recommender-System](https://github.com/kishan0725/Content-Based-Movie-Recommender-System) [3.b]
* [DeepLearningMovies](https://github.com/wendykan/DeepLearningMovies) [3.c]
  + [The-Movie-Cinema](https://github.com/kishan0725/The-Movie-Cinema) [3.a]
  + [Content-Based-Movie-Recommender-System](https://github.com/kishan0725/Content-Based-Movie-Recommender-System) [3.b]
  + [DeepLearningMovies](https://github.com/wendykan/DeepLearningMovies) [3.c]

Отличным источником датасетов является [Kaggle](https://www.kaggle.com) [4]. В частности, можно найти следующие датасеты:

* [Wikipedia Movie Plots](https://www.kaggle.com/jrobischon/wikipedia-movie-plots?select=wiki_movie_plots_deduped.csv) [4.a]

Датасет содержит 35000 строк. Среди полей можно найти такие как: Movie title, Origin of movie (i.e. American, Bollywood, Tamil, etc.), Director(s). Main actors/actresses, Movie genre(s), URL of Wikipedia page from which the plot description was scraped, Long form description of the movie's plot. Особо интересны здесь поля режиссер, жанр и основные актеры, которых нет в нашем датасете. В целом отдельную рекомендательную сеть было бы интересно сделать исключительно на полях актеров с учетом жанра и режиссера.

* [The Movies Dataset](https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv) [4.b]

Содержит 45000 строк. Список полей: adult, belongs\_to\_collection, budget, genres, homepage, id, imdb\_id, original\_language, original\_title, overview, popularity, poster\_path, production\_companies, production\_countries, calendar\_todayrelease\_date, revenue, runtime, spoken\_languages, status, tagline, title, checkvideo, vote\_average, vote\_count. Здесь особо интересны поля бюджет для каждого фильма, жанр, оригинальный язык, индекс популярности, производитель, выручка, длительность, средняя оценка, количество проголосовавших и некоторые другие.

В некоторых других датасетах так же встречается такие поля как автор/писатель, звезда (наиболее известный актер данного фильма) и другие не менее интересные для использования наборы данных.

Вообще на данной площадке очень много датасетов и запрос <https://www.kaggle.com/search?q=movie> [4.c] выдает 1256 датасетов. Остается только выбрать.

Так же на Kaggle.com интересны следующие соревнования, где можно посмотреть способы решения похожих задач, например, [https://www.kaggle.com/c/word2vec-nlp-tutorial](https://www.kaggle.com/c/word2vec-nlp-tutorialВ). Всего же по запросу <https://www.kaggle.com/search?q=movie+in%3Acompetitions> находятся 211 различных соревнований.

Кроме того, интересные датасеты можно найти на площадке <https://data.world> [5]. Например:

* [Wikipedia Movie Data from 1970-2018](https://data.world/typhon/wikipedia-movie-data-from-1970-2018) [5.a];
* [Movies](https://data.world/jamesgaskin/movies) [5.b];
* и другие.

Так же в процессе изучения данных были найдены уже построенные рекомендательные системы для фильмов, выведенные в продуктив:

* <https://movierecommendersystem.herokuapp.com> (на основании <https://github.com/kishan0725/Content-Based-Movie-Recommender-System>);
* <https://the-movie-buff.herokuapp.com> (на основании <https://github.com/kishan0725/The-Movie-Cinema>);
* <https://smhxrecommendersystemapp.herokuapp.com>;
* <https://serene-garden-31695.herokuapp.com/movies/>.

В целом можно сказать, что если бы не заранее зафиксированное в задании указание строить рекомендательную систему конкретно по исходящим ссылкам статьи описания фильма в Википедии, можно было бы построить не менее интересные рекомендательные системы для фильмов на основании других значимых признаков, таких, например, как режиссер, актеры, средняя оценка, бюджет, выручка, жанр и другие.

# Часть 3. Анализ и обработка используемых данных

Как было сказано выше, в работе был использован готовый датасет, предоставленный на курсе.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Name 10000 non-null object

1 Description 10000 non-null object

2 Links 10000 non-null object

3 Rating1 5584 non-null object

4 Rating2 5584 non-null object

dtypes: object(5)

memory usage: 390.8+ KB

В нем мы видим 5 полей: название фильма, его описание, ссылки со страницы фильма на другие страницы в Wiki, и два столбца с рейтингами. Далее посмотрим сами столбцы более подробно.

Поле Description. Здесь хранятся ключевые поля фильма, например:

'image': 'Deadpool poster.jpg',

'name': 'Deadpool',

'cinematography': 'Ken Seng',

'Software Used': 'Adobe Premier Pro',

'alt': "Official poster shows the titular hero Deadpool standing in front of the viewers, with hugging his hands, and donning his traditional black and red suit and mask, and the film's name, credits and billing below him.",

'distributor': '20th Century Fox',

'caption': 'Theatrical release poster',

'gross': '$783.1 million',

'country': 'United States',

'director': 'Tim Miller',

'runtime': '108 minutes',

'editing': 'Julian Clarke',

'language': 'English',

'music': 'Tom Holkenborg',

'budget': '$58 million'}

В частности, здесь можно найти бюджет и выручку фильма, язык, режиссера, длительность и многие другие интересные параметры. Однако важно отметить, что здесь жестко зафиксированных параметров, которые были бы у всех фильмах, и в разных строках о фильмах встречаются разные параметры. К наиболее часто встречающимся из них можно отнести:

[('name', 9948),

('director', 9764),

('image', 9649),

('language', 9607),

('runtime', 9314),

('country', 9169),

('cinematography', 8955),

('caption', 8757),

('music', 8710),

('distributor', 8561),

('editing', 8381),

('producer', 7273),

('gross', 6955),

('budget', 5875),

('writer', 5681)]

Это затрудняет обработку данных по данным параметрам. Поскольку если использовать, например, бюджет или выручку, то половина из фильмов будет потеряна, поскольку там нет данного значения.

Если говорить о паре ключ-значение, то в описании фильмов чаще всего встречаются следующие параметры:

[(('language', 'English'), 6683),

(('country', 'United States'), 5240),

(('caption', 'Theatrical release poster'), 5112),

(('country', 'India'), 1446),

(('language', 'Hindi'), 652),

(('country', 'United Kingdom'), 478),

(('distributor', 'Universal Pictures'), 455),

(('distributor', '20th Century Fox'), 453),

(('border', 'yes'), 407),

(('distributor', 'Paramount Pictures'), 405)]

Т.е. в принципе у нас очень много похожих фильмов по языку, стране и производителю.

В целом можно попробовать сделать рекомендательную систему на основании ключевых признаков, встречающихся в поле Description, и один из режимов модели будет реализован как раз для этого. Кроме того, можно выбрать конкретные ключевые признаки, которые нас интересуют, и построить модель именно по ним (в тестовой экспериментальной версии было реализовано, но в продуктив не пошло). Однако в задании конкретно написано использовать ссылки, поэтому рекомендательную систему по свойствам фильмов оставим на другой раз.

Если говорить о поле Links, ссылок в статье Википедии на другие ее же статьи, на основании которых и стояла задача строить модель, то мы увидим огромное количество самых различных ссылок на страницы самых разных типов:



В среднем по всему датасету в данное поле записано 114 ссылки на другие статьи.

Наиболее частое количество различных ссылок в данном поле – от 60 до 90:

|  |  |
| --- | --- |
|  | [(90, 113),  (64, 110),  (78, 106),  (85, 105),  (62, 104),  (66, 102),  (75, 101),  (70, 99),  (73, 98),  (77, 98),  (79, 97),  (56, 97),  (80, 97),  (67, 97),  (89, 94)] |

При этом видно, что часть из данных ссылок повторяются. При этом наиболее встречающимися являются следующие значения:

[('Rotten Tomatoes', 9393),

('Category:English-language films', 5882),

('Category:American films', 5867),

('Variety (magazine)', 5450),

('Metacritic', 5112),

('Box Office Mojo', 4186),

('The New York Times', 3818),

('The Hollywood Reporter', 3553),

('Roger Ebert', 2707),

('Los Angeles Times', 2454),

('Entertainment Weekly', 2375),

('British Board of Film Classification', 2236),

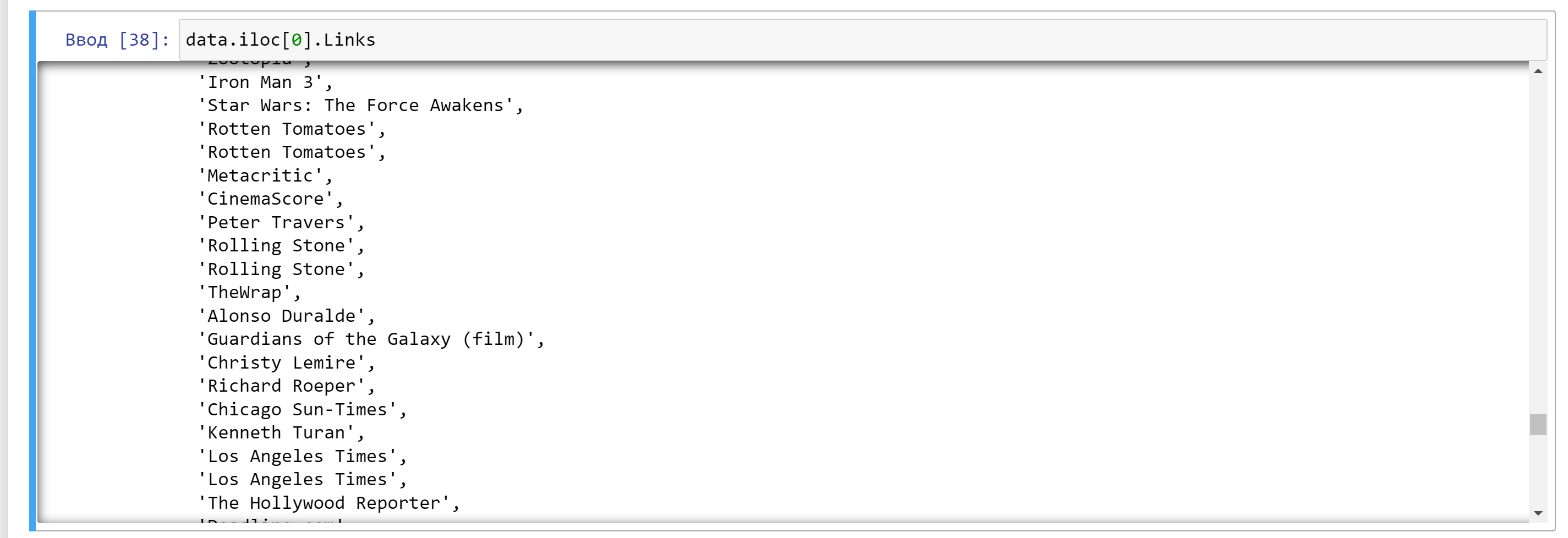
('Chicago Sun-Times', 1826),

('Deadline.com', 1814),

('The Guardian', 1528)]

Как видим, очень много ссылок на категории с англоязычными и американскими фильмами. При этом практически столько же ссылок, сколько и фильмов в датасете, на Rotten Tomatoes — сайт-агрегатор рецензий, на котором собирают обзоры фильмов и сериалов из различных изданий, информацию о фильмах и новости кинематографа. Кроме того, часто ссылаются на Variety — ведущий американский еженедельник, освещающий события в мире шоу-бизнеса; Metacritic — англоязычный сайт-агрегатор, собирающий отзывы о музыкальных альбомах, играх, фильмах, телевизионных шоу, DVD-дисках и мобильных играх; и Box Office Mojo — веб-сайт, постоянно отслеживающий кассовые сборы от кинопроката.

При этом если посмотреть на сами значения поля, то мы видим многочисленные дубликаты ссылок (Rotten Tomatoes, Rolling Stone, Los Angeles Times). Это легко объясняется тем, что ссылаться на одну и ту же страницу внутри статьи можно несколько раз.



Если убрать из данного поля дубликаты ссылок, т.е. оставить только уникальные вхождения, то наиболее частые вхождения будут от 50 до 60 раз.

|  |  |
| --- | --- |
|  | [(54, 136),  (51, 135),  (52, 128),  (47, 127),  (61, 126),  (50, 125),  (53, 124),  (59, 124),  (57, 124),  (64, 120),  (58, 118),  (63, 118),  (68, 115),  (41, 113),  (56, 112)] |

После удаления дубликатов в числе самых популярных следующие статьи:

[('Rotten Tomatoes', 5970),

('Category:English-language films', 5882),

('Category:American films', 5867),

('Metacritic', 3061),

('Box Office Mojo', 2627),

('Variety (magazine)', 2490)]

Т.е. мы видим, что на ‘Variety (magazine)’ в рамках одной и той же статьи зачастую ссылаются несколько раз, в результате после удаления дубликатов он существенно просел (примерно в два раз по количеству ссылок).

Если смотреть на поле ссылок Links дальше, то, к сожалению, данные ссылки сильно не структурированы и в них намешано все подряд из разных сфер и тематик. Однако отдельно можно выделить категории, к которым принадлежит данная статья (начинаются с ‘Category:’). И это будет еще одно новое поле, по которому можно вести анализ.

['Category:20th Century Fox films',

'Category:2010s action films',

'Category:2010s comedy films',

'Category:2010s superhero films',

'Category:2016 films',

'Category:American action comedy films',

'Category:American black comedy films',

'Category:American films',

'Category:Deadpool']

Кроме того, среди ссылок можно выделить те, которые совпадают с названиями имеющихся у нас фильмов, т.е. по сути идет ссылка на один из фильмов датасета, и сделать из них еще одно поле. К сожалению, треть от полученных значений будут пустые. Однако при этом интересно, что чаще всего ссылаются на Аватар, Титаник и Звездные воины. Впрочем, их отрыв от остальных не настолько велик.

('Avatar (2009 film)', 96),

('Titanic (1997 film)', 94),

'Star Wars (film)', 91),

'Jaws (film)', 84),

'The Godfather', 78),

'The Lion King', 76),

'Psycho (1960 film)', 72),

'The Dark Knight (film)', 66),

'Alien (film)', 65),

'Gone with the Wind (film)', 64),

'Citizen Kane', 62),

'The Matrix', 62),

'The Lord of the Rings (film series)', 58),

'Jurassic Park (film)', 54),

'Batman (1989 film)', 54),

'The Wizard of Oz (1939 film)', 53),

'Beauty and the Beast (1991 film)', 51),

'The Avengers (2012 film)', 50),

'Blade Runner', 50),

'Halloween (1978 film)', 50),

'E.T. the Extra-Terrestrial', 48),

'The Exorcist (film)', 46),

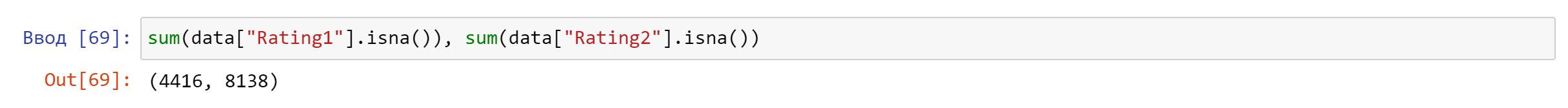
'The Shining (film)', 45),

'Batman Begins', 44),

'Snow White and the Seven Dwarfs (1937 film)', 44)]

Таким образом, на основании имеющегося поля со ссылками мы сделали несколько новых полей, по которым так же можно построить модель.

Кроме того, у нас есть два поля с оценками. При этом 4416 оценок из первого поля являются None. Из второго поля оценок количество пустых значений или None составляет 8138 из 10000.



Тем не менее, данные оценки можно применить для выбора лучших и худших фильмов. У нас две разные оценки, и не всегда ясно, на какую из них ориентироваться. Кроме того, первая и вторая оценки у нас даны в разных шкалах (из 10 или из 5). Простыми преобразованиями преобразуем оценки к единообразному числовому виду и создадим ‘балансовую’ третью оценку простым перемножением данных двух.

В качестве ключевого вывода из данной части можно сказать, что разная предобработка входящих данных и разное использование данных даст разные модели и результаты. В коде будет использоваться специальный параметр work\_option, отвечающий за выбор данных, по которым будем считать и строить модель.

# Часть 4. Построение модели выбора ближайших фильмов

Исходя из описания задачи требуется построить модель на основании данных о ссылках между статьями. Для этого проведем обучение эмбеддингов – векторных представлений данных.

Исходящие ссылки в поле Links можно трактовать как элементы взаимосвязи между статьями. При этом будем считать, что между фильмами, ссылки которых ведут на одну и ту же страницу, есть что-то общее. В процессе тренировке модель будет обучаться сходству не только между фильмами, а и между ссылками.

Анализ и предобработка ссылок Links были описаны в предыдущем разделе. После предобработки ссылок есть несколько различных наборов, на основании которых можно строить модель. Для выбора конкретного набора используется параметр work\_option, принимающий следующие значения:

{1: 'Работа по ссылкам (поле Links), дубликаты ссылок оставляем',

2: 'Работа по ссылкам (поле Links), дубликаты убираем',

3: 'Работа по описанию (поле Description)',

4: 'На основании ссылок (поле Links) на фильмы из датасета',

5: 'На основании ссылок на категории'}

По умолчанию будет использоваться параметр 1 - 'Работа по ссылкам (поле Links), дубликаты ссылок оставляем', поскольку именно это и требовалось исходя из задачи. Однако при желании его можно изменить и посмотреть работу модели по другим данным.

Для дальнейшего обучения модели выберем ссылки, которые встречаются наиболее часто, выкинув при этом низкочастотные. Т.е. фактически оставим только те ссылки, частота встречаемости которых не меньше некоторого параметра. Это служит как для ускорения работы модели (существенного по сравнению с тем, если брать вообще все ссылки), так и для изменения ее точности. Тестовый анализ показал, что относительно оптимальными в данном случае являются значения около min\_count\_links = 10.

Для более удобной работы с данными проиндексируем фильмы и имеющиеся ссылки, т.е. приведем их в числовой код. И составим два набора данных – фильмы и имеющиеся в них ссылки, приведя это к массиву кортежей пара: ссылка – фильм (т.е. номер ссылки и номер фильма, в котором встречается данная ссылка). При параметре min\_counts\_links, равном 10, у нас остаются исходные 10000 фильмов, 19126 различных ссылкок и 505031 пар соответствий между ними.

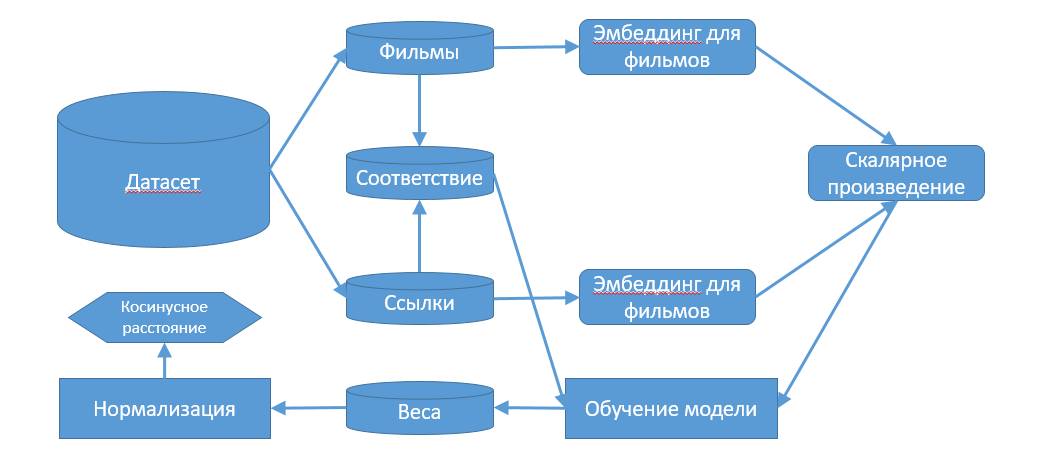
Таким образом, теперь у нас есть проиндексированные фильмы и проиндексированные ссылки. Далее с помощью библиотеки *keras* построим саму модель. В модели в качестве входного слоя будут передаваться тензоры единичной размерности. Для выходного построим эмбеддинги векторов ссылок и фильмов. Размерность эмбеддингов снова укажем в качестве параметра – *embedding\_size* – количество элементов в отображающем векторе – в данном случае выберем 100. Посчитаем скалярное произведение эмбеддингов для передачи на выход модели.

Передача данных в модель будет осуществляться с помощью генератора, который создает пакеты, состоящие из положительных и отрицательных примеров. Положительные примеры – это те, которые имеются в построенном множестве кортежей фильм-ссылка, отрицательные – те, которых нет в данном множестве. Количество положительных и отрицательных примеров в пакете так же задаются параметрами и могут регулироваться пользователем. В результате генератор будет генерировать кортеж фильм-ссылка и соответствующая метка (+-1), обозначающая наличие или отсутствие связи между элементами, который мы и будем подавать на вход модели.

Ожидается, что построенная модель будет обучаться весам так, чтобы сделать значение данного скалярного произведения произведения данных эмбеддингов как можно более близким к данной метке, т.е. по сути угадывать, есть между ними связь или нет. Получаемые веса модели будут проецировать фильмы и ссылки на такое пространство, в котором похожие фильмы будут находиться неподалеку друг от друга.

Далее обучаем построенную модель, где еще одним пользовательским параметром является количество эпох обучения. Будем задавать такое количество эпох, при котором на каждой следующей итерации функция потерь снижается уже не сильно.

Далее получим веса внутреннего слоя построенной модели обученной модели и нормализуем их для того, чтобы можно было использовать скалярное произведение в качестве меры близости.



Собственно, модель уже построена. Теперь можно считать близость между различными фильмами. Для этого достаточно посчитать скалярное произведение между построенными нормализованными векторами сравниваемых фильмов. Чем ближе результаты будут к 1, тем ближе будут сравниваемые фильмы между собой.

Например, показатели близости для 'Harry Potter (film series)':

33 - Harry Potter (film series) --- 0.0

871 - Harry Potter and the Half-Blood Prince (film) --- 0.0123

827 - Harry Potter and the Order of the Phoenix (film) --- 0.024

332 - Harry Potter and the Philosopher's Stone (film) --- 0.0243

677 - Harry Potter and the Goblet of Fire (film) --- 0.0265

828 - Harry Potter and the Prisoner of Azkaban (film) --- 0.0275

380 - Harry Potter and the Deathly Hallows – Part 2 --- 0.0332

698 - Harry Potter and the Deathly Hallows – Part 1 --- 0.0333

967 - Harry Potter and the Chamber of Secrets (film) --- 0.043

169 - Alice in Wonderland (2010 film) --- 0.071

62 - Fantastic Beasts and Where to Find Them (film) --- 0.0879

56 - Pirates of the Caribbean (film series) --- 0.1065

1092 - Lemony Snicket's A Series of Unfortunate Events --- 0.1152

91 - The Hobbit (film series) --- 0.1303

688 - Stardust (2007 film) --- 0.1326

И это был один из тех фильмов, по которому шла отладка модели и коррекция имеющихся параметров. После отладки хорошую близость показывают и другие похожие друг на друга фильмы, которые в отладке не участвовали. Например:

416 - Spider-Man (2002 film) --- 0.0

784 - Spider-Man 2 --- 0.0303

478 - Spider-Man 3 --- 0.036

613 - Terminator Salvation --- 0.0528

448 - Minority Report (film) --- 0.0611

1085 - Planet of the Apes (2001 film) --- 0.0644

1159 - Cowboys & Aliens --- 0.0646

118 - Watchmen (film) --- 0.0666

214 - Transformers (film) --- 0.0667

61 - Man of Steel (film) --- 0.0727

Или:

155 - Gladiator (2000 film) --- 0.0

260 - Braveheart --- 0.0469

666 - Valkyrie (film) --- 0.051

35 - Titanic (1997 film) --- 0.0567

669 - Munich (film) --- 0.058

702 - Robin Hood (2010 film) --- 0.0582

1121 - Empire of the Sun (film) --- 0.0583

634 - The Patriot (2000 film) --- 0.0673

2343 - United 93 (film) --- 0.0702

1128 - The Phantom of the Opera (2004 film) --- 0.0729

Таким образом, можно считать, что результаты получились достаточно хорошие, и данную модель можно использовать для дальнейшей работы.

Далее с помощью классификатора требуется построить рекомендательную систему, чтобы давать рекомендации о фильмах.

# Часть 5. Построение модели рекомендаций по интересам пользователя или набору выбранных фильмов

Для построения рекомендательной системы на основе множества интересных пользователю фильмов будем использовать классификатор. Выбор классификатора в данной работе так же зададим через отдельный параметр class\_option\_description. По умолчанию будем использовать метод линейный опорных векторов и далее описывать построение модели именно с его помощью.

Общая суть метода опорных векторов состоит в построении такой разделяющей гиперплоскости между наборами данных, которая максимизирует расстояние между различными классами, а точнее, расстояние между гиперплоскостью и ближайшими к ней точками, называемыми опорными векторами. Расстояние между гиперплоскостью и опорными векторами называется зазором. Лучшей будет являться та модель, для которой данный зазор окажется максимально большим. В случае использования линейного ядра данная гиперплоскость будет линейной.

В качестве входных данных данному классификатору подадим наборы лучших и худших фильмов. Это могут быть те фильмы, которые пользователь оценил как хорошие и плохие соответственно, и тогда у нас получится классическая рекомендательная система на основе предпочтений пользователя. Или те, которые мы сами выбрали по некоторым критериям. Или их можно определить на основе имеющегося у нас рейтинга. Ну и так далее.

Для данной работы в качестве лучших возьмем известные фильмы, собравшие большие бюджеты и пользующиеся большой популярностью, такие как Titanic (1997 film)', 'Avatar (2009 film)', 'Gladiator (2000 film)' и другие; в качестве худших будем брать худшие фильмы по имеющимся у нас оценкам.

После запуска и выполнения модели получаем следующие результаты:

8 Star Wars (film) 1.3964264059204956

35 Titanic (1997 film) 1.3961574376674382

303 Raiders of the Lost Ark 1.3380092309698173

84 Saving Private Ryan 1.3076145665976293

134 Citizen Kane 1.303469488242184

172 Schindler's List 1.2935488825074482

220 Jaws (film) 1.2614526611603203

129 Gone with the Wind (film) 1.2447629414938972

310 Ben-Hur (1959 film) 1.2351620640359018

155 Gladiator (2000 film) 1.2269492558346862

66 Skyfall 1.2226541042815045

Таким образом, по входящему набору хороших и плохих фильмов (некоторого абстрактного пользователя) с помощью датасета данных, содержащего ссылки на релевантные страницы, мы смогли найти выбрать другие подходящие для данного пользователя фильмы. В целом полученные результаты вполне релевантны ожидаемым, что говорит о достаточно хорошем качестве построенной модели.

# Часть 6. Результаты применения моделей и методов

Таким образом, в рамках данной работы мы построили две модели. Первая модель оценивает близость фильмов и может найти наиболее близкие фильмы к указанному нами. Вторая модель основывается на первой и позволяет по набору лучших и худших фильмов найти те, которые понравятся пользователю больше всего.

Фактически результаты построенных моделей очень сильно зависят от передаваемых параметров. В данной работе сознательно не была реализована фиксация параметров и параметры остались задаваемыми с той целью, чтобы пользователь мог сам поиграться и найти наиболее подходящую для него выдачу результатов. Кроме того, понятие близости фильмов относительное, и по результатам работы, фильмы, получившие наибольшую близость в построенных моделях с точки зрения косинусного расстояния, далеко не всегда оказывались близкими по смыслу. В целом в качестве развития модели можно предварительно экспертно задать модели более близкие по смыслу фильмы с тем, чтобы подбирать параметры автоматически, минимизируя отличие результатов модели с экспертно заданными показателями.

Исходя из используемых параметров:

rnd\_seed = 17

work\_option = 1

min\_count\_links = 10

embedding\_size = 100

nn\_metric = 'mse'

positive\_samples\_per\_batch = 256

negative\_ratio = 16

epochs = 35

наиболее близкие по смыслу к фильмам о Гарри Поттере стали фильмы о Гарри Поттере, а Человек Паук стал близок к фильмам о Человеке Пауке (в некоторых случаях несмотря на гораздо меньшую ошибку все было совершенно иначе).

В общем-то это можно уже использовать в качестве рекомендаций по типу «Если Вам интересен фильм A, обратите внимание на фильм B». Для этого была разработана отдельная функция, которая выбирает пользователю конкретный подходящий фильм, с вероятностью, в той или иной степени пропорциональной его потенциальному интересу к данному фильму (т.е. более лучший фильм будет показываться существенно чаще, чем тот, который проигрывает по рейтингу; чем больше основание степени, еще одного параметра, – тем чаще).

Далее c помощью классификатора на основании предоставленного списка лучших и худших фильмов для конкретного пользователя был разработан рейтинг наиболее подходящих фильмов. В зависимости от используемых параметров и задаваемого изначального списка фильмов это будут разные фильмы.

Построенную систему можно развивать далее и при выборе использовать, например, жанры фильма, рейтинги и оценки, а так же многие другие параметры; фиксированно или с некоторыми весами. А так же строить гибридные модели и прочие надстройки. В зависимости от потребностей заказчика и необходимости использования данной системы.

# Заключение

Таким образом, была построена модель выбора наиболее подходящих фильмов. При этом в качестве данных для построения модели использовались ссылки описания предложенных фильмов в Википедии.

Методами автоматического перебора параметров были получены результаты с очень близкими значениями векторов «ближайших» фильмов и низкой ошибкой соответствующей построенной модели, однако на практике ближайшие с точки зрения построенных таким образом моделей фильмы не были близки с точки зрения смысла и их реального содержания, поэтому в итоге параметры выбирались с учетом экспертной оценки, насколько полученные результаты в итоге адекватны и релевантны тому, что действительно нужно. И это, наверное, один из основных выводов данной работы – что полученные с точки зрения косинусной близости результаты были не настолько близки с точки зрения реального смысла, и лучшими оказались те, у которых и ошибка модели и косинусная близость были несколько больше.

Теоретически в качестве продолжения можно изначально вручную сопоставить близкие фильмы, которые должны бы получиться близкими, и тестировать модель и параметры уже на данном сопоставлении, смотреть, насколько это действительно так, и выбирать параметры исходя из наибольшего соответствия тому, что «должно быть» (и тогда это уже будет обучение с учителем, пусть и на начальном этапе учительства), но по причине ограниченности ресурсов и субъективности данной оценки близости данная часть работы оставлена на будущее развитие данной модели.

Кроме того, в качестве продолжения можно использовать многочисленные альтернативные значимые поля данных и датасетов, а не только исходящие ссылки, тем самым уточняя, улучшая модель, а так же строя многочисленные гибридные модели рекомендательной системы.

# Библиографический список

1. [Recommendation Systems — Models and Evaluation](https://towardsdatascience.com/recommendation-systems-models-and-evaluation) – towardsdatascience.com;

1. [Алгоритмы формирования рекомендательных сервисов](https://telesputnik.ru/materials/tech/article/algoritmy-formirovaniya-rekomendatelnykh-servis/) - telesputnik.ru;
2. Д. Осинга – Глубокое обучение: Готовые решения. “Диалектика”, Санкт-Петербург, 2020 г.

1. [Рекомендательные системы - Прикладные задачи анализа данных](https://www.coursera.org/learn/data-analysis-applications/lecture/saLcw/riekomiendatiel-nyie-sistiemy) - coursera.org – Московский физико-технический институт, Яндекс

1. [Я.С. Погорельская - Обзор подходов к построению рекомендательных систем](https://events.rudn.ru/event/107/papers/511/files/866-%D0%9F%D0%BE%D0%B3%D0%BE%D1%80%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F.pdf) – Российский университет дружбы народов.

# Приложение 1. Датасеты для построения рекомендательной системы по фильмам

1. <https://dumps.wikimedia.org>
   1. <https://dumps.wikimedia.org/enwiki>
   2. <https://dumps.wikimedia.org/ruwiki>
2. [Wikipedia-film-database](https://github.com/piki/wikipedia-film-database)
   1. [oracleofbacon.org/data.txt.bz2](https://oracleofbacon.org/data.txt.bz2)
3. [Movies-dataset](https://github.com/topics/movies-dataset)
   1. [The-Movie-Cinema](https://github.com/kishan0725/The-Movie-Cinema)
   2. [Content-Based-Movie-Recommender-System](https://github.com/kishan0725/Content-Based-Movie-Recommender-System)
   3. [DeepLearningMovies](https://github.com/wendykan/DeepLearningMovies)
   4. [Film-Release-Date-Analysis](https://github.com/mspamelalea/Film-Release-Date-Analysis)
4. [Kaggle](https://www.kaggle.com)
   1. [Wikipedia Movie Plots](https://www.kaggle.com/jrobischon/wikipedia-movie-plots?select=wiki_movie_plots_deduped.csv)
   2. [The Movies Dataset](https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv)
   3. [Search by movie](https://www.kaggle.com/search?q=movie)
5. [data.world](https://data.world)
   1. [Wikipedia Movie Data from 1970-2018](https://data.world/typhon/wikipedia-movie-data-from-1970-2018)
   2. [Movies](https://data.world/jamesgaskin/movies)

1. [Датасеты от АНО ВО «Университет Иннополис»](https://drive.google.com/drive/folders/1r9KZPqUBcuDnyvNoJCFfEd4RJY1Z2jXL)