МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СЕРВИСА РЕКОМЕНДАЦИЙ КНИГ НА ЯЗЫКЕ PYTHON

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Наконечной Веры Викторовны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня системы рекомендаций широко используются для рекомендации продуктов пользователям на основе их интересов. Система рекомендаций - одна из сильнейших систем для увеличения прибыли за счет удержания большего числа пользователей в условиях жесткой конкуренции.

Онлайн-сервисы для просмотра фильмов или прослушивания музыки, такие как "Кинопоиск"или "Окко"конкурируют друг с другом по многим факторам. Одним из таких важных факторов является их РС. Цель системы – спрогнозировать интерес покупателя и соответственно рекомендовать ему именно то, что он захочет приобрести. Такие системы могут учитывать множество параметров, таких как жанр и содержание, путем фильтрации отзывов пользователей, статистики покупок и т.д. они составляют для каждого пользователя индивидуальную подборку рекомендаций. Это достигается с помощью прогностического моделирования и эвристики с использованием доступных данных.

Сейчас работа над изучением и улучшением работы рекомендательных систем является активно развивающейся научной отраслью, над которой трудятся не только коммерческие компании, но и многие научные деятели.

Как выяснили ученые, избыток выбора увеличивает когнитивную нагрузку на мозг. А по оценкам психолога Барри Шварца, разнообразие парализует волю и делает человека несчастнее. В то время как рекомендательные системы способны облегчить выбор, а значит помогают человеку стать счастливее.

Рекомендательные системы оказались полезными и в медицине. К примеру, они могут находить новые соединения активных веществ для лекарств, а также находить новые свойства у клинически одобренных лекарств.

Для исследования тематики машинного обучения и алгоритмов работы рекомендательных систем, было принято решение разработать сервис рекомендаций книг на языке Python.

Был поставлен ряд задач:

- 1. изучить алгоритм работы рекомендательных сервисов;
- 2. отобрать данные для обучения модели;
- 3. создать сервис для рекомендаций книг.

1 Изучение алгоритмов работы рекомендательных сервисов

1.1 Типы рекомендательных систем

Существует 3 основных методики рекомендаций:

1.1.1 Фильтрация на основе контента

Методы фильтрации на основе контента основаны на описании продукта и профиле предпочтений пользователя.

Системы, основанные на контенте, используют информацию о характеристиках и учитывают атрибуты элементов. Существует много методов извлечения признаков. Можно создать вектор слов из описания элемента или из слов, собранных в соответствии с предпочтениями пользователя.

Одна из возникающих проблем заключается в предоставлении очевидных рекомендаций из-за чрезмерной специализации.

1.1.2 Совместная фильтрация

Метод совместной фильтрации основан на сборе и анализе данных о поведении пользователя. Сюда входит его онлайн-активность и прогнозирование того, что ему понравится, на основе сходства с другими пользователями.

Одним из основных преимуществ этой РС является то, что она может точно рекомендовать сложные элементы, не разбираясь в самом объекте. Нет никакой зависимости от содержимого, поддающегося машинному анализу. С математической точки зрения, используется "Матрица полезности".

Этот подход наиболее полезен, когда есть тонна данных, и они имеют высокую разреженность. Матричная факторизация помогает РС путем понижения размерности, что ускоряет вычисления. Такой вариант идеально подходит для разрабатываемой РС, так как будет очень большое количество данных, как о пользователях, так и о книгах.

1.1.3 Гибридные рекомендательные системы

В гибридных системах рекомендаций для продуктов рекомендуется использовать фильтрацию на основе контента и совместную фильтрацию одновременно. Эта система рекомендаций находится на стадии разработки и развивается в данный момент компаниями-гигантами. Она также может наложить вето на общие проблемы в системах рекомендаций, такие как проблемы с холодным запуском и недостаточностью данных.

1.2 Алгоритм Nearest Neighbors

Для обучения модели и дальнейшей качественной выдаче рекомендаций был выбран алгоритм – Nearest Neighbors или "Ближайшие соседи".

Принцип, лежащий в основе методов "ближайшего соседа состоит в том, чтобы найти предопределенное количество обучающих выборок, ближайших по расстоянию к новой точке, и предсказать метку по ним. Количество выборок может быть заданной пользователем константой (обучение k-ближайшего соседа) или изменяться в зависимости от локальной плотности точек (обучение соседей на основе радиуса). Расстояние, как правило, может быть любой метрической мерой. В разрабатываемой системе по рекомендации книг, константа будет задана вручную для корректной работы сервиса. Для вычисления "ближайшего соседа"будет использоваться косинусное сходство.

1.2.1 Типы алгоритма

NearestNeighbors реализует обучение ближайших соседей без учителя. Он действует как единый интерфейс к трем различным алгоритмам ближайших соседей: BallTree, KDTree и алгоритму перебора Brute.

1.2.2 Выбор подходящего типа

Так как в разрабатываемой рекомендательной системе есть потребность в определении более 1 "ближайшего соседа". Время запроса Ball tree и KD tree будет медленнее, при увеличении K. Это связано с двумя эффектами: во-первых, большие K приводят к необходимости поиска в большей части пространства параметров. Во-вторых, при K > 1 требуется внутренняя организация очереди результатов (по мере обхода дерева).

В ситуации с подготовленными данными и K = 6, наилучшим решением будет сделать кластеризацию при помощи метода Brute с дополнительной метрикой - поиском косинусного расстояния, для повышения точности и улучшения работы PC.

2 Разработка сервиса

2.1 Подбор данных

Для разработки полноценной РС были отобраны и выгружены с сайта https://www.amazon.com несколько пакетов данных.

Данные о 271360 книгах были собраны в файле BX-Books.csv.

Данные о 278858 пользователях были собраны в файле BX-Users.csv.

Данные о 1149780 рейтингах книг были собраны в файле BX-Book-Ratings.csv.

2.2 Обработка данных

2.2.1 Отсечение неактивных пользователей

Оценив полученные данные, можно сделать вывод, что есть много пользователей, которые оценили много книг. Пользователи с малым количеством книг не дадут нужный результат при обучении модели, т.к. их рекомендации будут чрезмерно специфическими.

Для дальнейшей работы, были отфильтрованы только те пользователи, которые оценили более 100 книг.

```
x = ratings['user_id'].value_counts() > 100
y = x[x].index
ratings = ratings[ratings['user_id'].isin(y)]
```

Листинг 1: Код для отбора и выделения пользователей в новую группу

Таким образом отобраны 899 пользователей.

2.2.2 Отсечение книг на основании оценок

Для того чтобы обработать данные о книгах, был написан следующий код, который агрегирует данные о книгах и их рейтинге:

```
ratings_with_books = ratings.merge(books, on='ISBN')
```

Листинг 2: Код для просмотра рейтинга по книгам

Чтобы узнать количество оценок по каждой книге, был написан код:

```
number_rating = ratings_with_books.groupby('title')['rating'].count().
    reset_index()
number_rating.rename(columns={'rating':'num_of_rating'},inplace=True)
```

Листинг 3: Код для просмотра количества оценок по книгам

Для рекомендаций были отобраны те книги, по которым есть более 50 оценок. Такие книги будут с большей вероятностью иметь честную среднюю оценку, также можно будет составить приблизительный портрет их читателей.

```
final_rating = ratings_with_books.merge(number_rating, on='title')
final_rating = final_rating[final_rating['num_of_rating'] >= 50]
```

Листинг 4: Код для отбора популярных книг

Таким образом для дальнейшей работы были отобраны 59850 книг.

2.2.3 Создание сводной таблицы

Для продолжения работы была создана сводная таблица данных – матрица, которая содержит данные о пользователях и книгах, а на пересечении о оценках. Таким образом можно будет выделить пользователей с похожими предпочтениями в особые группы – кластеры.

```
book_pivot = final_rating.pivot_table(columns='user_id', index='title', values=
    'rating')
```

Листинг 5: Код для создания сводной таблицы

Для того чтобы работа с данными была качественной и 0.0 в матрице не замедляли работу, была подключена библиотека scipy.sparse для обработки метрик и получения рассредоточенной матрицы.

Таким образом была получена матрица 742 на 888 из ранее отобранных данных, которая будет использоваться для обучения модели.

2.3 Обучение модели

Для того чтобы обучить модель на отобранных данных, необходимо импортировать выбранный ранее алгоритм KNN:

```
from sklearn.neighbors import NearestNeighbors
model = NearestNeighbors(metric = 'cosine', algorithm= 'brute')
model.fit(book_sparse)
```

Листинг 6: Код для импортирования алгоритма кластеризации

Данный алгоритм будет возвращать 2 параметра: Расстояние и Рекомендация.

Допустим алгоритм "Ближайших соседей"будет запущен для следующих параметров:

```
distance, suggestion = model.kneighbors(book_pivot.iloc[237,:].values.reshape
    (1,-1), n_neighbors=6 )
```

Листинг 7: Код для подбора рекомендаций ко второй книге о Гарри Поттере

В данном случае будет возвращено distance для 6 "соседей"выбранной книги, где 0 – расстояние до неё самой и suggestion – рекомендуемые книги.

Чтобы перевести эти данные в удобный для человека формат, был написан следующий код:

```
def recommend_book(book_name):
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.
   reshape(1,-1), n_neighbors=6 )

    for i in range(len(suggestion)):
            books = book_pivot.index[suggestion[i]]
            for j in books:
                if j == book_name:
                    print(f"You searched '{book_name}'\n")
                    print("The suggestion books are: \n")
                else: print(j)
```

Листинг 8: Функция выдачи рекомендаций

Чтобы убедиться в том, что данный код работает корректно, было проведено 2 тестовых запуска:

```
book_name = "Harry Potter and the Chamber of Secrets (Book 2)"
recommend_book(book_name)

You searched 'Harry Potter and the Chamber of Secrets (Book 2)'

The suggestion books are:

Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Goblet of Fire (Book 4)
Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
Harry Potter and the Sorcerer's Stone (Book 1)
Harry Potter and the Order of the Phoenix (Book 5)
```

Рисунок 1 – Тестовый запуск №1

В выдаче для "Гарри Поттера"рекомендуются книги из этой же серии.

Если выбрать другую книгу, которая не включена в какую либо серию, можно увидеть следующие результаты:

```
book_name = "Angels &amp; Demons"
recommend_book(book_name)
```

```
You searched 'Angels &amp; Demons'

The suggestion books are:

The Da Vinci Code
Bel Canto: A Novel
The Sweet Potato Queens' Book of Love
Daisy Fay and the Miracle Man
A Wrinkle in Time
```

Рисунок 2 – Тестовый запуск №2

Таким образом можно сделать вывод что РС работает успешно, т.к. наиболее похожими на выбранную книгу, она рекомендовала еще 3 известные книги того же автора и 2 титулованные книги других авторов.

Чтобы окончательно убедиться в том, что был выбран верный тип алгоритма KNN, были проведены ещё 2 запуска для KD tree и Ball tree. Их выдача оказалась идентичной:

```
book_name = "Harry Potter and the Chamber of Secrets (Book 2)"
recommend_book(book_name)
```

```
You searched 'Harry Potter and the Chamber of Secrets (Book 2)'

The suggestion books are:

Harry Potter and the Goblet of Fire (Book 4)
Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Sorcerer's Stone (Book 1)
The Charm School
The Road to Omaha
```

```
book_name = "Angels &amp; Demons"
recommend_book(book_name)
```

```
You searched 'Angels &amp; Demons'

The suggestion books are:

Personal Injuries
No Safe Place
The Cat Who Went into the Closet
Tom Clancy's Op-Center: Games of State (Tom Clancy's Op Center (Paperback))
Ground Zero and Beyond
```

Рисунок 3 – Выдача для KD tree и Ball tree

Видно, что данные рекомендации подходят к введённой книге хуже, чем при алгоритме Brute, хотя скорость обработки данных равна. Таким образом можно сделать вывод, что в ходе исследования алгоритмов и методик обработки данных, был сделан правильный выбор.

2.4 Реализация Web-приложения

Для быстрого и простого создания Web-сервиса по рекомендации книг был использован streamlit. Streamlit – это веб-фреймворк, предназначенный для простого развертывания моделей и визуализаций с использованием Python.

Для того чтобы сформировать локальную страничку, достаточно импортировать streamlit. Для того, чтобы пользователь мог вводить названия книг вручную или выбирать из списка, был написан следующий код:

```
selected_books = st.selectbox(
    "Type or select a book from the dropdown",
    book_names)
```

Листинг 9: Код для выбора книги из списка

Для подготовки страницы к показу рекомендаций был написан следующий код, который создаёт 5 колонок с местом под текст – название книги и под изображение – обложку книги:

```
if st.button('Show Recommendation'):
    recommended_books,poster_url = recommend_book(selected_books)
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(recommended_books[1])
        st.image(poster_url[1])
    .....
```

Листинг 10: Код для подготовки страницы к выводу рекомендаций

Чтобы выводить на экран подобранные рекомендованные книги в формате "Название + Обложка был написан следующий код:

```
def fetch_poster(suggestion):
    book_name = []
    ids_index = []
    poster_url = []

    for book_id in suggestion:
        book_name.append(book_pivot.index[book_id])
    for name in book_name[0]:
        ids = np.where(final_rating['title'] == name)[0][0]
        ids_index.append(ids)
    for idx in ids_index:
        url = final_rating.iloc[idx]['image_url']
        poster_url.append(url)
    return poster_url
```

Листинг 11: Код для вывода названий и обложек рекомендованных книг

Итоговая функция, которая будет принимать наименование книги от пользователя, обрабатывать его при помощи ML и выдавать список рекомендаций на экран, была написана следующим образом:

```
def recommend_book(book_name):
    books_list = []
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(book_pivot.iloc[book_id,:].values.
   reshape(1,-1), n_neighbors=6 )
    poster_url = fetch_poster(suggestion)
    for i in range(len(suggestion)):
            books = book_pivot.index[suggestion[i]]
            for j in books: books_list.append(j)
    return books_list , poster_url
```

Листинг 12: Итоговая функция Web-сервиса рекомендаций книг

Таким образом при запуске сервиса, выборе необходимой книги и нажатии на кнопку поиска, при помощи ML будут подобраны рекомендации, на основании данных о сотнях книг, данных других пользователей и их личных оценок книг:

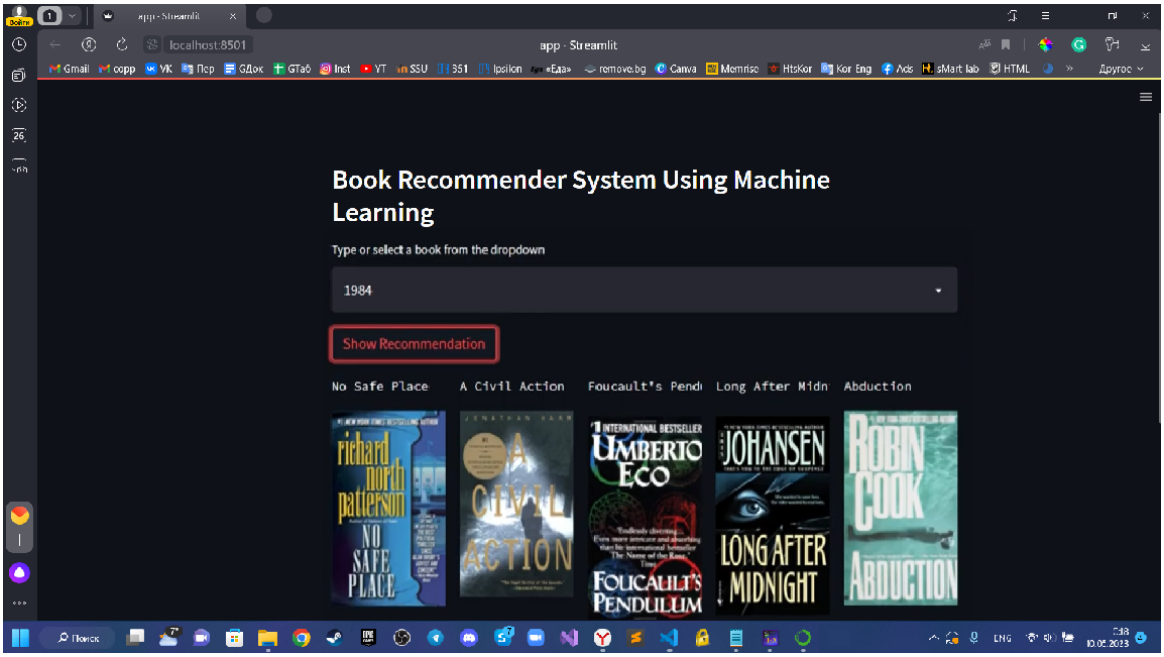

Рисунок 4 – Сервис рекомендаций книг в работе

ЗАКЛЮЧЕНИЕ

Системы рекомендаций приобретают все большее значение в современном чрезвычайно загруженном мире. Людям всегда не хватает времени для решения множества задач, которые им необходимо выполнить за ограниченные 24 часа. Именно поэтому системы рекомендаций важны, поскольку они помогают пользователям делать правильный выбор, не затрачивая свои когнитивные ресурсы.

Была проделана большая практическая работа. Была создана рекомендательная система, которая на вход получает лишь название книги, но возвращает 5 хороших книг. Это возможно благодаря развитию компьютерных наук и машинному обучения. Любой человек может воспользоваться данной системой и найти для себя новую интересную книгу.