УДК 004.9

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ: ОБЗОР И АНАЛИЗ МЕТОДОВ

Ларин С.Э., студент гр. Б-ИСиТ-22, II курс
Научный руководитель: Белаш В.Ю., к.пед.н, доцент
Калужский государственный университет им. К. Э. Циолковского
г. Калуга

Практическая значимость научной статьи определяется избытком информации в различных формах – фильмы, сериалы, музыка, книги, что создает проблему выбора для пользователя, поскольку сложно соответствовать именно тому, что удовлетворяет его конкретным предпочтениям.

Рекомендательные системы анализируют предпочтения каждого пользователя и предлагают контент, который будет интересен именно ему с высокой вероятностью.

Главной целью таких систем является оптимизация навигации пользователя по информационному пространству, а также поиск более релевантных вариантов.

Наиболее популярным методом рекомендательных систем является метод коллаборативной фильтрации (КФ) – это технология, которая позволяет прогнозировать предпочтения конкретного пользователя интернет-ресурса, сравнивая его интересы с интересами других посетителей ресурса [3].

На рисунке 1 представлена классификация рекомендательных систем.

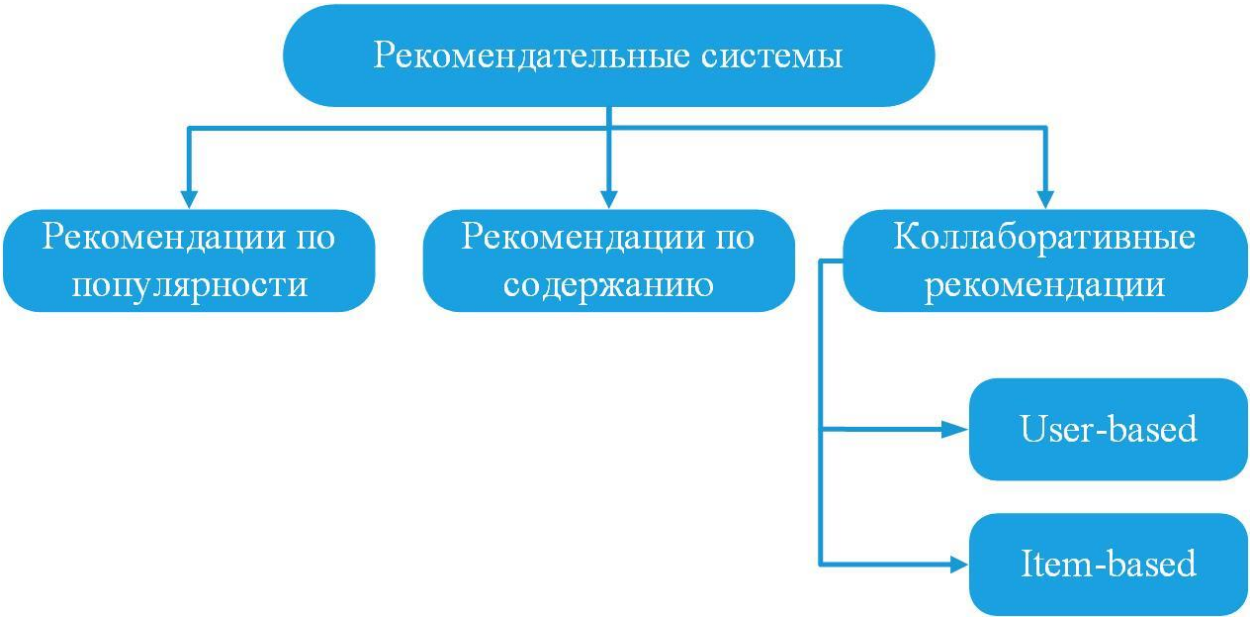

Рисунок 1 – Типы рекомендательных систем

Рекомендации по популярности являются самым простым типом рекомендательных систем. Они основаны на предположении – чем выше рейтинг товара, тем выше вероятность рекомендации. Недостатком является отсутствие учета индивидуальных предпочтений конкретного пользователя.

Рекомендации по содержанию анализируют потребляемый контент и предлагают наиболее похожий контент. Например, если пользователю понравился фильм с жанром фантастики, то скорее всего система будет рекомендовать похожие фильмы с аналогичным жанром. Основным недостатком является требование в более качественном контенте с подробным описанием.

Коллаборативные рекомендации (КР) основаны на анализе поведения других пользователей. КР подразделяется на два основных метода: User-based и Item-based. При методе User-based происходит поиск пользователей, имеющих общие понравившиеся объекты. Item-based находит объекты, похожие на те, которые уже понравились данному пользователю.

Перед тем как переходить к изучению КФ, необходимо ознакомиться с общей схемой работы рекомендательных систем, которая представлена на рисунке 2.

Кандидаты (candidates): пользователи и объекты могут исчисляться миллионами, поэтому необходим отбор кандидатов. Из огромной базы данных отбирается небольшой список кандидатов, как правило от нескольких сотен до тысячи. В группу кандидатов входят фильмы, книги, товары, то есть объекты, которые могут быть рекомендованы пользователю.

В ранжировании (ranking) система учитывает факторы по степени их соответствия предпочтениям пользователя.

Бизнес-правила (business rules) – особый набор критериев, которые определяют ограничения и фильтруют рекомендации. Например, не рекомендовать контент, который не подходит по возрасту пользователя.

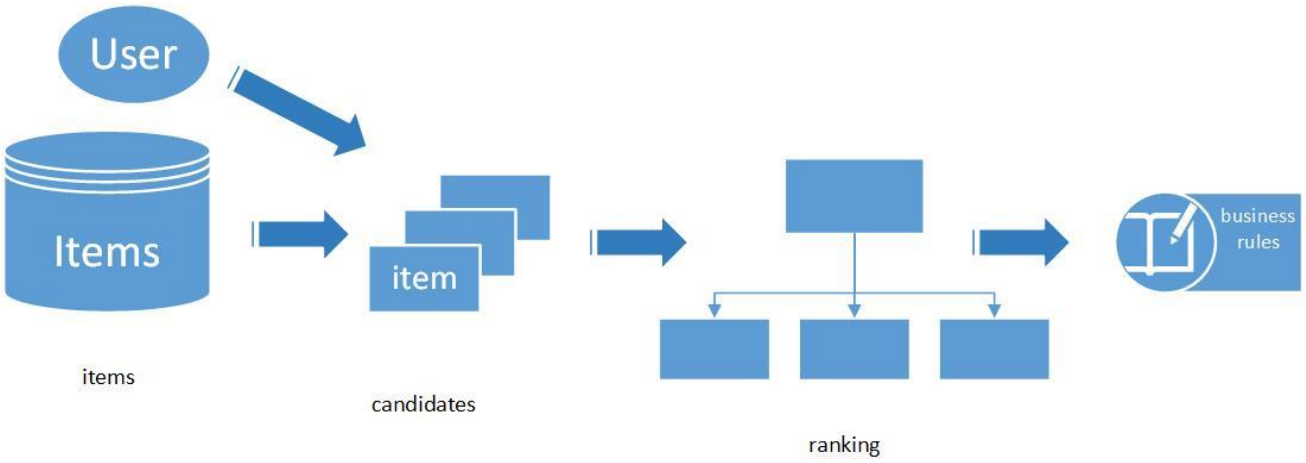

Рисунок 2 – Общая схема рекомендательных систем

После рассмотрения общей схемы рекомендательных систем перейдем к подробному изучению КФ.

Для демонстрации работы алгоритма КФ воспользуемся урезанным датасетом с оценками пользователей для фильмов – MovieLens. Данный датасет является классическим примером для оценки качества рекомендательных систем.

Рассмотрим два типа алгоритмов КФ: user-item filtering и item-item filtering.

1. Последовательность действий алгоритма User-item filtering выглядит следующим образом:

Выбирается исходный элемент (фильм, музыка, товар).

Осуществляется поиск пользователей, которым нравится этот элемент.

Анализируется история оценок пользователей, которым нравится данный элемент.

Формируется рекомендация на основе анализа истории оценок.

Для реализации метода были использованы следующие шаги:

- Загрузка данных: Данные о рейтингах фильмов загружаются из CSV-файлов.
- Создание матрицы оценок: Матрица оценок создается с фильмами в строках и пользователями в столбцах. Ячейки матрицы содержат оценки пользователей для соответствующих фильмов.
- Обработка пропусков: Пропуски в матрице оценок заполняются средним значением по столбцу (т. е. средним рейтингом фильма).
- Расчет косинусного сходства: Косинусное сходство между фильмами вычисляется с использованием функции cosine_similarity из библиотеки scikit-learn.
- Рекомендации: для заданного фильма находятся 5 наиболее похожих фильмов с использованием косинусного сходства.

Листинг программы на Python:

```
movie_id = 1  # "Toy Story (1995)"
recommendations = item_based_recommendations(movie_id)
print(f"Пользователям, которым нравится
\"{movies_df.iloc[movie_id]['title']}\", может так же понравиться:")
for movie in recommendations:
  print(f"- {movie}")
```

Переменной movie_id присваивается значение 1, что соответствует фильму "Toy Story (1995)". recommendations = item_based_recommendations(movie_id):

Функция item_based_recommendations вызывается с аргументом movie_id.

Функция возвращает список из 5 фильмов, наиболее похожих на "Toy Story (1995)".

print(f"Пользователям, которым нравится \"{movies_df.iloc[movie_id]['title']}\", может так же понравиться:"):

Выводятся заголовок"Пользователям, которым нравится "Toy Story (1995)", может так же понравиться:".

for movie in recommendations:

Перебираются все фильмы в списке recommendations.

Пример работы User-item представлен на рисунке 3.

```
Пользователям, которым нравится "Jumanji (1995)", может так же понравиться:
- Children of the Corn IV: The Gathering (1996)
- Rescuers, The (1977)
- Grand Budapest Hotel, The (2014)
- Big Empty, The (2003)
- In the Name of the King: A Dungeon Siege Tale (2008)
```

Рисунок 3 – Список из пяти рекомендаций для фильма Jumanji

2. Item-item filtering:

Данные о предпочтениях пользователей обычно хранятся в виде матрицы user-item, где строки представляют пользователей, столбцы – элементы, а значения на пересечении строки и столбца - оценки, поставленные пользователем конкретному элементу.

Вычисляется схожесть элементов (косинусное сходство, коэффициент Жакарда, коэффициент корреляции Пирсона) [2].

Формируется рекомендация на основе вычисленных значений схожести элементов.

К сожалению, недостаток данных в датасете фильмов ограничивает возможности метода item-based filtering. Небольшое количество фильмов является препятствием по нахождению пользователей с похожими предпочтениями, что негативно влияет на точность и эффективность рекомендаций или вовсе не выдает результата, как видно из рисунка 4.

Данный метод использует следующие шаги:

- Загрузка данных: Данные о рейтингах фильмов загружаются из CSV-файлов.
- Создание объекта KNN: создается объект KNN с метрикой косинусного сходства и алгоритмом "brute force".
- Обучение модели: Модель обучается на данных о рейтингах фильмов.
- Рекомендации: для заданного пользователя находятся 5 пользователей с наиболее схожими рейтингами. На основе фильмов, которые понравились этим пользователям, формируется список рекомендаций.

Листинг программы на Python:

```
# Поиск похожих пользователей
  try:
    # Извлечение рейтингов пользователя по ID
    user_ratings = ratings_df[ratings_df['userId'] == user_id]['rating'].values.reshape(1, -1)
    similar_users = knn.kneighbors(user_ratings, n_neighbors=6, return_distance=False)
  except ValueError:
```

```
            return None
        # Получение рекомендаций на основе фильмов, оцененных похожими пользователями
            recommended_movies = []
            for similar_user in similar_users[0][1:]:
                # Извлечение ID фильмов, оцененных похожими пользователями
                movie_ids = ratings_df[ratings_df['userId'] == similar_user]['movieId'].values
                # Извлечение названий фильмов
            recommended_movies.extend(movies_df[movies_df['movieId'].isin(movie_ids)]['title'].tolist())
            recommendations = user_based_recommendations(user_id):
            Функция user_based_recommendations
            вызывается с аргументом user_id.
            Функция возвращает список из 5 фильмов, которые нравятся пользователям, схожим с пользователем с ID user_id.
            if recommendations is not None: Если функция user_based_recommendations возвратила непустой список (т.е. удалось найти похожих пользователей), то выполняется следующий код:
            print(f"Похожим на вас пользователям нравится:"): Выводятся слова "Похожим на вас пользователям нравится:".
            for movie in recommendations: Перебираются все фильмы в списке recommendations.
            print(f"- {movie}"): Для каждого фильма выводится его название с префиксом "-".
            else:: Если функция user_based_recommendations возвратила пустой список (т.е. не удалось найти похожих пользователей), то выводится сообщение "Не удалось найти похожих пользователей для ID {user_id}".
```

```
Не удалось найти похожих пользователей для ID 1
```

Рисунок 4 – Получение рекомендаций на основе похожих пользователей

Метод коллаборативной фильтрации является одним из наиболее популярных для рекомендательных систем. Он обладает рядом преимуществ, таких как:

1. Эффективность – система способна рекомендовать контент, который действительно интересен пользователю.
2. Простота – метод легко реализуется и интерпретируется.
3. Рекомендательные системы, основанные на методе КФ, являются эффективным инструментом для персонализации контента.

В будущем можно ожидать построение более точных и персонализированных рекомендаций: системы будут учитывать не только историю действий пользователя, но и его контекст, намерения.

Список литературы:

1. ИТМО // Рекомендательные системы URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B (дата обращения: 05.03.2024).
2. Кокачев В.А. Рекомендательные системы в контексте технологий больших данных: дис. Фундаментальная информатика и информационные технологии наук: Спб, 2018. - 41 с.
3. Коллаборативная фильтрация простыми словами // Лалалань URL: https://lala.lanbook.com/kollaborativnaya-filtraciya-prostymi-slovami (дата обращения: 20.03.2024).