

УДК 004.8

Сравнительный анализ методов объединения в гибридных рекомендательных системах

Гусько Глеб Александрович

ka-s-2015@mail.ru

Михайлов Алексей Сергеевич

mikhailow.aleksei2018@yandex.ru

Вершинин Евгений Владимирович

vershinin@bmstu.ru

КФ МГТУ им. Н.Э. Баумана, Калуга, Россия

Данное исследование включает разработку и анализ эффективной рекомендательной системы, внедряемой в веб-приложение по заказу еды, основанной на гибридном подходе, который сочетает контентную и коллаборативную фильтрации. Подробно рассматриваются преимущества и недостатки различных методов объединения рекомендаций, таких как взвешенное комбинирование, каскадный, ансамблевый и многоуровневый подходы. В настоящем исследовании особое внимание уделяется выбору оптимальной стратегии, которая позволит повысить точность рекомендаций.

Ключевые слова: гибридный метод, фильтрация, контентная фильтрация, коллаборативная фильтрация, методы объединения в гибридной фильтрации

С развитием рынка услуг, который стремительно расширяется и адаптируется к изменяющимся потребностям пользователей, персонализированные рекомендации становятся одним из важнейших инструментов для повышения вовлеченности клиентов и увеличения конверсии. Хорошим примером стремительной адаптации к потребностям пользователя может стать бизнес, связанный с кафе и ресторанами, которые работают на доставку.

Персонализация в сфере доставки еды позволит клиентам быстрее находить то, что им действительно нравится. Это, в свою очередь, поспособствует увеличению числа повторных заказов и общей оценки сервиса. Для достижения этой цели необходимо использовать эффективные методы фильтрации, которые позволяют анализировать данные о предпочтениях пользователей.

Создание эффективной рекомендательной системы для веб-приложения по заказу еды требует комплексного подхода, который учитывает, как индивидуальные предпочтения пользователей, так и общие тенденции в потребительском поведении. Это не только повысит удовлетворенность клиентов, но и поможет компаниям увеличить свою долю на конкурентном рынке доставки еды.

Целью настоящего исследования является выявление оптимального метода объединения контентной и коллаборативной фильтраций для рекомендательной системы в веб-приложении по заказу еды.

Обзор методов объединения рекомендаций. Создание эффективных рекомендательных систем — сложная задача, требующая учета множества факторов, включая индивидуальные предпочтения пользователей, динамику изме-

нения их вкусов и огромные объемы данных. Поэтому разрабатываются гибридные подходы, комбинирующие преимущества различных методов. Существует несколько основных архитектур таких систем, каждая из которых имеет свои сильные и слабые стороны, рассмотрим их.

1. Метод комбинирования результатов. В его основе лежит идея объединения прогнозов, полученных от нескольких независимых моделей. Например, одна модель может предсказывать предпочтения на основе истории просмотров пользователя, другая — на основе информации о его демографических данных, а третья — на основе анализа поведения похожих пользователей (collaborative filtering). Затем результаты этих моделей агрегируются. Простейший способ — это простое арифметическое усреднение оценок. Однако более эффективный подход использует взвешенное суммирование, где каждой модели присваивается вес, отражающий ее точность и надежность.

Веса могут быть определены эмпирически, например, на основе результатов кросс-вариации на исторических данных, или же с помощью более сложных методов, например, используя байесовские подходы или методы оптимизации, такие как градиентный спуск. Важно отметить, что при таком подходе необходимо тщательно отслеживать вклад каждой модели, чтобы избежать ситуации, когда одна доминирующая модель подавляет другие, потенциально более полезные. Более того, для повышения эффективности можно использовать нелинейные методы агрегации, например нейронные сети, которые могут обучаться оптимальному способу объединения прогнозов.

Преимущества:

- гибкость: можно настроить веса для каждого метода (коллаборативной и контентной фильтраций), чтобы отдавать предпочтения конкретной модели в зависимости от цели системы. Это позволяет адаптировать алгоритм под разные условия;

- скорость обработки: объединение рекомендаций с помощью взвешенного суммирования, снижает время обработки рекомендаций на 30 % по сравнению с методами многоуровневых ансамблей [1]. Это важно для приложений с высокими требованиями к скорости отклика, например, в мобильных приложениях, где пользователи ожидают время ответа не более 100–150 мс;

- точность: объединение контентных и коллаборативных подходов приводит к улучшению метрики NDCG (Normalized Discounted Cumulative Gain) на 10–15 % и повышению метрики MAP (Mean Average Precision) на 12 % в сравнение с использованием одной модели. Это указывает на то, что комбинирование методов снижает недостатки отдельных подходов, что особенно полезно для рекомендаций с учетом сезонных изменений или временных предпочтений пользователей [1].

Недостатки:

- сложность настройки весов: подбор оптимальных весов для объединения рекомендаций может быть трудоемким процессом. Настройка весов через кросс-валидацию добавила до 25 % к общему времени на обучение, а неверные веса привели к снижению точности рекомендаций на 10 %. В связи

с этим требуется постоянная переоценка качества весов, что может стоить большого количества ресурсов [1];

– нагрузка: высокая нагрузка на центральный процессор из-за параллельной работы нескольких моделей.

2. Каскадный метод. Каскадный метод предполагает последовательное применение нескольких моделей, где выход одной модели является входными данными для следующей. Это позволяет эффективно обрабатывать данные, сначала отсеивая нерелевантные варианты, а затем более детально анализируя оставшиеся. Например, первый уровень может быть основан на быстрой, но не очень точной модели, которая отфильтрует большинство неинтересных вариантов. Следующий уровень может использовать более сложную и ресурсоемкую модель, но только для меньшего подмножества предложенных вариантов. Выбор модели на каждом этапе может зависеть от характеристик пользователя или контекста запроса. К примеру, для новых пользователей, у которых нет истории просмотров, можно использовать модель, основанную на популярности товаров, в то время как для опытных пользователей — более персонализированную модель, учитывающую их прошлые действия. Это позволяет оптимизировать скорость работы системы без потери качества рекомендаций.

Преимущества:

– повышенная точность: каскадный метод может улучшить конечную точность рекомендаций за счет последовательной фильтрации данных. Каскадная система улучшила точность на 18 % по метрике Precision и на 15 % по метрике Recall, относительно системы с использованием одной модели, так как более детальные модели обрабатывают уже отфильтрованные данные [2];

– эффективное использование моделей: поскольку каждая модель работает с уже отфильтрованными данными, каскадный метод значительно экономит вычислительные ресурсы. Данный подход снижает потребление ресурсов на 35 % за счет предварительной фильтрации данных более быстрой моделью на первом уровне, что особенно важно для больших систем рекомендаций [2];

– улучшенная персонализация: в каскадном подходе каждый следующий этап все лучше адаптируется к запросам пользователя. По метрике NDCCG, данный подход позволил улучшить персонализацию на 20 % за счет применения адаптированных моделей для старых и новых пользователей [2];

– гибкость: в зависимости от задачи можно адаптировать последовательность каскада: начинать с простой модели и постепенно добавлять более сложные. Такая конфигурация может быть легко изменена для достижения оптимальных результатов.

Недостатки:

– задержка в скорости рекомендаций: последовательная обработка каскадных моделей может замедлить генерацию рекомендаций. Время отклика

системы при использовании каскадного метода увеличилось на 40 % по сравнению с одной моделью, что требует более тщательной оптимизации для приложений, чувствительных к времени отклика [3];

- возможность накопления ошибок: ошибки на начальном уровне каскада могут передаваться на последующие уровни, ограничивая качество итоговой рекомендации. Если первая модель ошибочно отсеет релевантные рекомендации, последующие уровни не смогут их вернуть;

- требования к большому объему данных: для эффективных результатов каскадного метода необходимо значительное количество данных на начальных уровнях. При ограниченном объеме данных точность конечных рекомендаций может снижаться до 20 % по сравнению с одноуровневыми методами, поскольку каждый следующий уровень нуждается в достаточном объеме данных для правильной работы [3]. В частности, исследования показывают, что на начальных этапах каскада требуется от 1000 до 5000 пользовательских взаимодействий на каждого уникального пользователя для получения устойчивых результатов.

3. Ансамблевое обучение. Ансамблевое обучение использует методы, такие как бустинг (например, Gradient Boosting Machines, XGBoost, LightGBM) или случайный лес (Random Forest), для объединения множества «слабых» моделей в одну более мощную. Эти методы создают множество различных моделей, обучаемых на случайных подмножествах данных или с использованием различных параметров. Затем предсказания этих моделей агрегируются, например, с помощью усреднения или голосования. Ансамблевые методы часто демонстрируют высокую точность, но могут быть более сложными в реализации и требуют больше вычислительных ресурсов.

Преимущества:

- высокая точность и устойчивость: ансамблевые методы показывают улучшение точности по сравнению с единичными моделями. Ансамблевое обучение с XGBoost улучшило точность классификации на 20 % по сравнению с использованием одной модели [4]. В то же время ансамбль из случайных лесов и градиентного бустинга показали себя хуже, с их помощью удалось увеличить точность на 12 %;

- снижение переобучения: ансамблевые методы, такие как бэггинг (например, случайный лес), снижают риск переобучения, так как обучаются на разных подмножествах данных [4]. Это делает модели более обобщенными и устойчивыми к шуму.

Недостатки:

- высокая вычислительная сложность: ансамбли, особенно сложные, такие как стекинг и бустинг, требуют значительных вычислительных ресурсов для обучения и предсказаний, что может замедлить обработку данных;

- сложность настройки: настройка ансамблевых моделей может быть сложной, так как необходимо выбрать и оптимизировать сразу несколько моделей. Это требует времени и экспертизы для выбора оптимальных параметров и комбинаций моделей;

– меньшая интерпретируемость: ансамблевые модели труднее интерпретировать, особенно если они включают сложные алгоритмы. Это может быть проблемой в областях, где требуется объяснять, почему были сделаны определенные предсказания;

– риск переобучения при неправильной настройке: плохо настроенные ансамбли могут переобучаться, несмотря на устойчивость к шуму. Использование чрезмерно глубоких деревьев в случайном лесу может увеличить ошибку в тестовых данных на 10 % по сравнению с правильно настроенными ансамблями, что подчеркивает важность их настройки для устранения переобучения [4];

– затрудненное обновление моделей: ансамблевые модели, такие как случайный лес, часто требуют полного переобучения для эффективного включения новых данных. Повторное обучение ансамблевой модели на новых данных может занять в 1,7 раза больше времени, чем добавление данных к единичной модели, что увеличивает операционные затраты на обслуживание модели [4].

4. Многоуровневый метод. Многоуровневый метод комбинирует преимущества разных методов на различных этапах процесса рекомендаций. Например, первый уровень может использовать контент based filtering для отбора наиболее релевантных товаров, а второй уровень — collaborative filtering для ранжирования этих товаров в соответствии с предпочтениями пользователя. В этом случае, первый уровень выполняет предварительную фильтрацию, уменьшая объем данных, которые обрабатывает более сложная и ресурсоемкая вторая модель. Это позволяет повысить как скорость работы, так и точность рекомендаций. Более того, каждый уровень может включать механизмы обратной связи, позволяющие уточнять прогнозы и адаптироваться к меняющимся предпочтениям пользователей.

Преимущества:

– гибкость в выборе моделей: на каждом уровне можно использовать разные алгоритмы в зависимости от задачи и требований. Это позволяет адаптировать подход под конкретную предметную область и данные;

– улучшенное качество рекомендаций: благодаря тому, что каждый уровень уточняет результаты предыдущего, итоговые рекомендации становятся более релевантными. Добавление коллаборативного фильтра на второй уровень после контентного фильтра увеличило точность рекомендаций на 12 %, что особенно важно в системах с большим количеством вариантов [5];

– эффективное распределение вычислительных ресурсов: начальные уровни могут использовать более простые алгоритмы, снижая общую нагрузку на систему. Многоуровневая система, использующая легкий контентный фильтр на первом уровне, снижает потребление вычислительных ресурсов на 30 % по сравнению с системой, которая сразу применяет сложные алгоритмы [5].

Недостатки:

– сложность настройки и разработки: многоуровневая система требует тщательно продуманной архитектуры и настройки, так как необходимо пра-

вильно выбирать алгоритмы и параметры для каждого уровня. Это усложняет процесс разработки и требует времени;

- задержки в скорости: прохождение рекомендаций через несколько уровней может возрасти задержка. Многоуровневая система может увеличить задержку на 25 %, особенно если на поздних уровнях используются ресурсоемкие модели, что требует дополнительной оптимизации для приложений [5];

- риск накопления ошибок: ошибки на одном уровне могут передаваться на следующие и влиять на качество итоговых рекомендаций. Если начальные уровни отсеют релевантные данные, более глубокие уровни не смогут их вернуть;

- высокие вычислительные затраты: несмотря на оптимизацию начальных уровней, общие вычислительные затраты могут быть высокими, особенно если на последних уровнях используются сложные модели. Данный подход увеличил использование ресурсов на 20 % по сравнению с одноуровневой системой при работе с большими объемами данных [5];

- сложность интерпретации результатов: многоуровневый подход затрудняет анализ и объяснение рекомендаций, так как каждый уровень вносит свои изменения.

После тщательного анализа различных методов на основе их преимуществ и недостатков, пришли к выводу, что наиболее подходящими являются методы взвешенного комбинирования и каскадный метод. Эти методы не только обеспечивают высокую скорость обработки данных, но и демонстрируют хорошую адаптивность к изменяющимся предпочтениям пользователей [6].

Реализация. Реализация алгоритмов рекомендаций включает следующие этапы.

1. Реализация контентной фильтрации.

Контентная фильтрация

```
def content_based_filtering(user_id):  
    cursor.execute("""  
        SELECT preferred_ingredients, preferred_cuisines  
        FROM user_preferences  
        WHERE user_id = ?  
        """, (user_id,))  
    preferences = cursor.fetchone()
```

```
    cursor.execute("""  
        SELECT item_id  
        FROM menu_items  
        WHERE ingredients LIKE ? OR cuisine IN (?)  
        ORDER BY popularity DESC  
        LIMIT 5  
        """, (f"%{preferences[0]}%", preferences[1]))  
    return [row[0] for row in cursor.fetchall()]
```

2. Реализация коллаборативной фильтрации.

Коллаборативная фильтрация с учетом сходства и предпочтений

```
def collaborative_filtering(user_id, max_recommendations=10, similarity_threshold=SIMILARITY_THRESHOLD):
```

```
# Получаем список похожих пользователей с учетом порога сходства
cursor.execute("""
SELECT other_user_id, similarity_score
FROM user_similarity
WHERE user_id = ?
AND similarity_score >= ?
ORDER BY similarity_score DESC
""", (user_id, similarity_threshold))
similar_users = cursor.fetchall()

if not similar_users:
    return None # Недостаточно данных для коллаборативной фильтрации

# Получаем заказы текущего пользователя
user_past_orders = get_user_past_orders(user_id)

# Получаем предпочтительные ингредиенты текущего пользователя
user_preferences = get_user_preferences(user_id)

recommendations = []
recommendation_count = {}

for user, similarity_score in similar_users:
    cursor.execute("""
SELECT item_id
FROM orders
WHERE user_id = ?
GROUP BY item_id
ORDER BY COUNT(*) DESC
""", (user,))
    user_recommendations = [row[0] for row in cursor.fetchall()]

    # Проверяем заказы похожих пользователей
    for item in user_recommendations:
        # Исключаем блюда, которые пользователь уже заказывал
        if item not in user_past_orders:
            # Проверяем, что блюдо соответствует предпочтениям по ингредиентам
            cursor.execute("""
SELECT ingredients
FROM menu_items
WHERE item_id = ?
""", (item,))
            ingredients = cursor.fetchone()[0].split(',')

            # Смотрим, чтобы хотя бы один ингредиент совпадал с предпочтениями пользователя
            if any(ingredient in user_preferences for ingredient in ingredients):
                # Учитываем сходство пользователя в весе рекомендации
                if item in recommendation_count:
                    recommendation_count[item] += similarity_score
                else:
                    recommendation_count[item] = similarity_score
```

```
# Сортируем блюда по весу (с учетом сходства) и ограничиваем количество реко-
мендаций
```

```
sorted_recommendations = sorted(recommendation_count.items(), key=lambda x:
x[1], reverse=True)
```

```
recommendations = [item[0] for item in sort-
ed_recommendations[:max_recommendations]]
```

```
return recommendations
```

3. Объединение методом каскадной фильтрации.

```
# Каскадная система рекомендаций
```

```
def get_recommendations(user_id):
```

```
# Пробуем коллаборативную фильтрацию
```

```
recommendations = collaborative_filtering(user_id)
```

```
# Если нет рекомендаций из коллаборативной фильтрации, используем
контентную
```

```
if not recommendations:
```

```
recommendations = content_based_filtering(user_id)
```

```
# Получаем названия блюд
```

```
if recommendations:
```

```
recommendation_names = get_item_names(recommendations)
```

```
else:
```

```
recommendation_names = []
```

```
# Вывод информации для пользователя
```

```
print(f"Каскадная рекомендация для пользователя с ID {user_id}:")
```

```
if recommendation_names:
```

```
for item_name in recommendation_names:
```

```
print(f"- {item_name}")
```

```
else:
```

```
print("Рекомендаций нет.")
```

4. Объединение взвешенным методом.

```
# Веса
```

```
# Веса для алгоритмов
```

```
COLLABORATIVE_WEIGHT = 5.0
```

```
CONTENT_WEIGHT = 5.0
```

```
# Взвешивание результатов
```

```
def weighted_combination(collaborative_recs, content_recs):
```

```
score_map = {}
```

```
# Присваиваем баллы коллаборативной фильтрации
```

```
if COLLABORATIVE_WEIGHT > 0:
```

```
for rec in collaborative_recs:
```

```
score_map[rec] = score_map.get(rec, 0) + COLLABORATIVE_WEIGHT
```

```
# Присваиваем баллы контентной фильтрации, только если вес > 0
```

```
if CONTENT_WEIGHT > 0:
```

```

for rec in content_recs:
    score_map[rec] = score_map.get(rec, 0) + CONTENT_WEIGHT

# Сортируем по весам (более высокие весовые баллы идут в начало)
return sorted(score_map, key=score_map.get, reverse=True)

# Функция для получения названия блюда по его ID
def get_item_name(item_id):
    cursor.execute("SELECT item_name FROM menu_items WHERE item_id = ?", (item_id,))
    result = cursor.fetchone()
    return result[0] if result else None

# Преобразование списка идентификаторов блюд в их названия
def convert_ids_to_names(item_ids):
    return [get_item_name(item_id) for item_id in item_ids if get_item_name(item_id)]

# Взвешенная система рекомендаций
def get_weighted_recommendations(user_id):
    # Получаем рекомендации от обоих методов
    collaborative_recs = collaborative_filtering(user_id) or []
    content_recs = content_based_filtering(user_id) or []

    # Преобразуем идентификаторы в названия
    collaborative_recs_names = convert_ids_to_names(collaborative_recs)
    content_recs_names = convert_ids_to_names(content_recs)

    # Выводим списки рекомендаций с названиями
    print("Коллаборативная: ", collaborative_recs_names)
    print("Контентная: ", content_recs_names)

    # Если ни один алгоритм не дал результатов, возвращаем пустой список
    if not collaborative_recs and not content_recs:
        return []

    # Комбинируем результаты
    final_recommendations = weighted_combination(collaborative_recs, content_recs)

    # Преобразуем идентификаторы блюд в их текстовые названия
    final_recommendation_names = convert_ids_to_names(final_recommendations)

    return final_recommendation_names

```

Полученные рекомендации показаны на рис. 1.

```

Каскадная рекомендация для пользователя с ID 7:
- Tacos
- Caesar Salad
Коллаборативная: ['Tacos', 'Caesar Salad']
Контентная: ['Hamburger', 'Veggie Burger']
Взвешенные рекомендации для пользователя 7: ['Tacos', 'Caesar Salad', 'Hamburger', 'Veggie Burger']

```

Рис. 1. Пример результатов гибридной фильтрации для пользователя с ID 7

Теперь проанализируем полученные данные для всех пользователей и построим графики для выбора наилучшего метода объединения используя следующие метрики [7].

Персонализация. Показатели персонализации каждого метода рассчитывались, основываясь на том, насколько точно рекомендации отражают предпочтения пользователей. Персонализация была измерена для каждого пользователя, основываясь на полученных рекомендациях. Высокие значения означают лучшее соответствие предпочтениям пользователя.

Сходство внутри списка. Этот показатель отражает степень, в которой элементы в одном рекомендательном списке схожи друг с другом. Значения были рассчитаны на основе анализа сходства объектов в рекомендациях. Более низкое значение сходства указывает на разнообразие, а более высокое — на узкую направленность рекомендаций.

Длинный хвост рекомендаций. Данные для этого графика отражают фактическое распределение популярности рекомендованных объектов среди пользователей, где каждый объект имеет количество упоминаний в рекомендациях.

1. Персонализация. На графике (рис. 2) показано, как варьируется уровень персонализации для каждого пользователя при использовании каскадного и взвешенного методов. Ось X представляет уникальных пользователей, а ось Y — уровень персонализации.

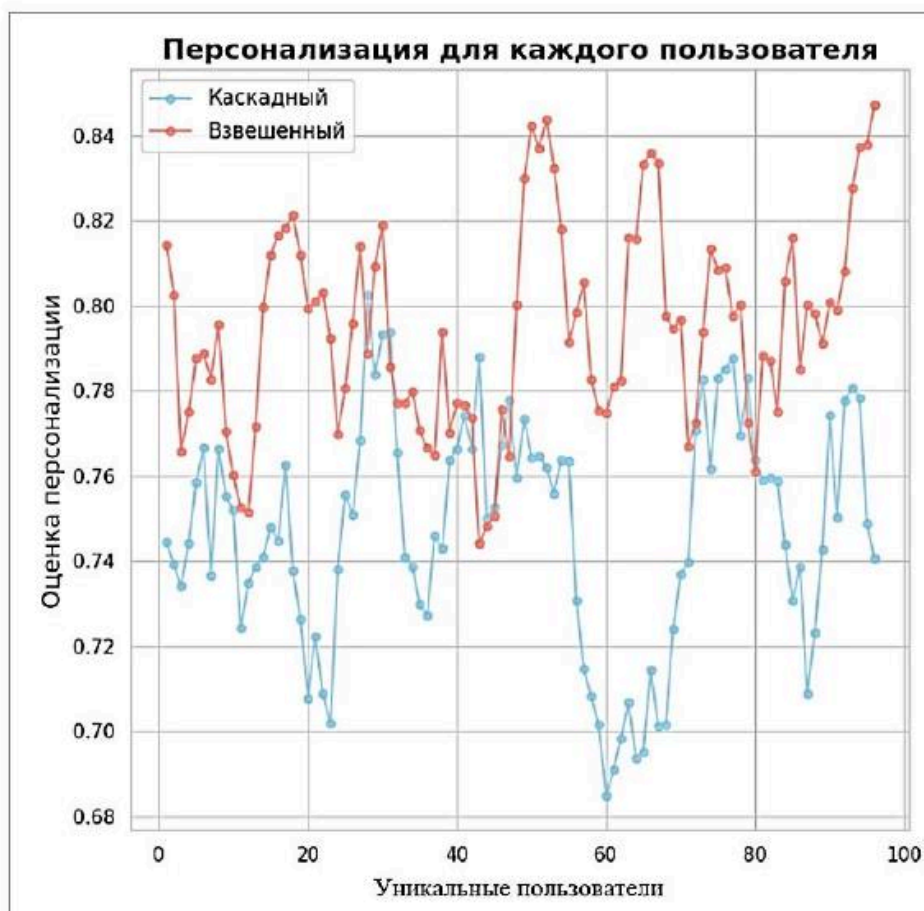


Рис. 2. Сравнение персонализации

Видно, что взвешенный метод демонстрирует более высокие показатели персонализации, что отражает его способность лучше учитывать предпочтения пользователей на основе данных, собранных за время их взаимодействия с системой. Каскадный метод в целом достигает более низкого уровня персонализации, поскольку комбинирует рекомендации упрощенно, следуя правилам последовательности. Это свидетельствует о том, что взвешенный метод более эффективен в адаптации рекомендаций под каждого конкретного пользователя.

2. Сходство внутри списка. На втором графике (рис. 3) сравнивается «внутрисписочное» сходство между каскадным и взвешенным методами, где ось X представляет пользователей, а ось Y — показатель внутрисписочного сходства.

Каскадный метод демонстрирует более высокие значения сходства, что указывает на меньшую диверсификацию рекомендаций. Это связано с тем, что данный метод чаще предлагает схожие объекты, которые уже доказали свою популярность среди пользователей с похожими предпочтениями. Взвешенный метод, напротив, показывает рекомендации с более низким внутрисписочным сходством, что делает его более разнообразным. Эти результаты свидетельствуют о том, что взвешенный метод лучше адаптируется к разнообразию предпочтений пользователей, в то время как каскадный метод склонен к предоставлению более узких, специализированных рекомендаций.

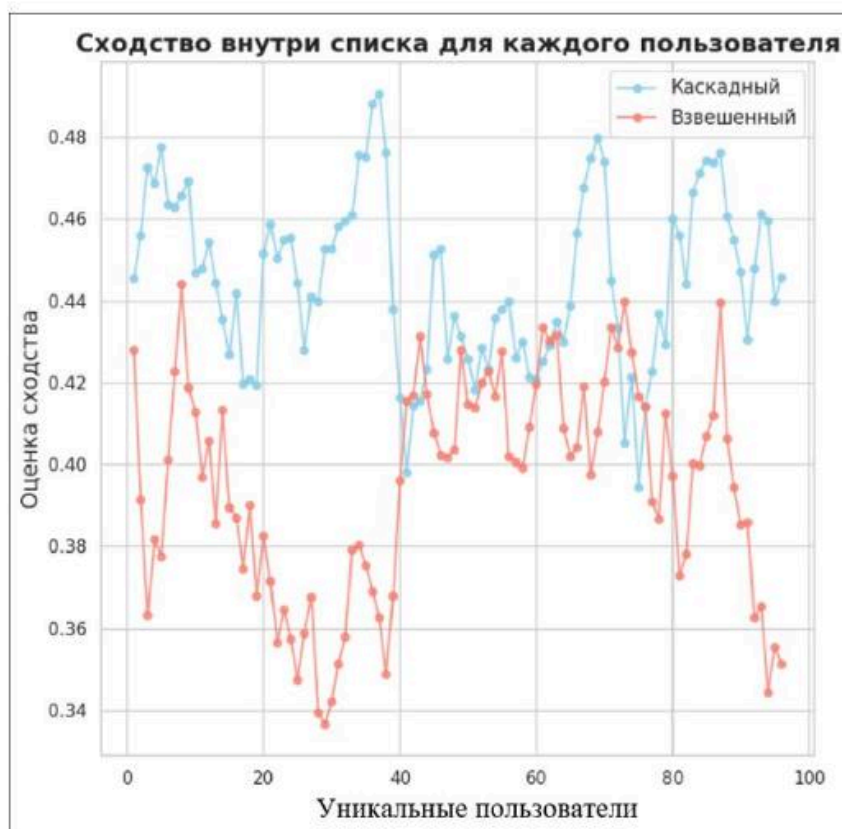


Рис. 3. Сравнение сходства

3. Длинный хвост. На третьем графике (рис. 4) сравнивается эффект «длинного хвоста» для каскадного и взвешенного методов рекомендаций. Ось X представляет ранжированные позиции рекомендаций для блюд (от наиболее к наименее популярным), а ось Y — количество рекомендаций. График иллюстрирует, как часто пользователи получают рекомендации на основе популярных или менее популярных объектов (блюд).

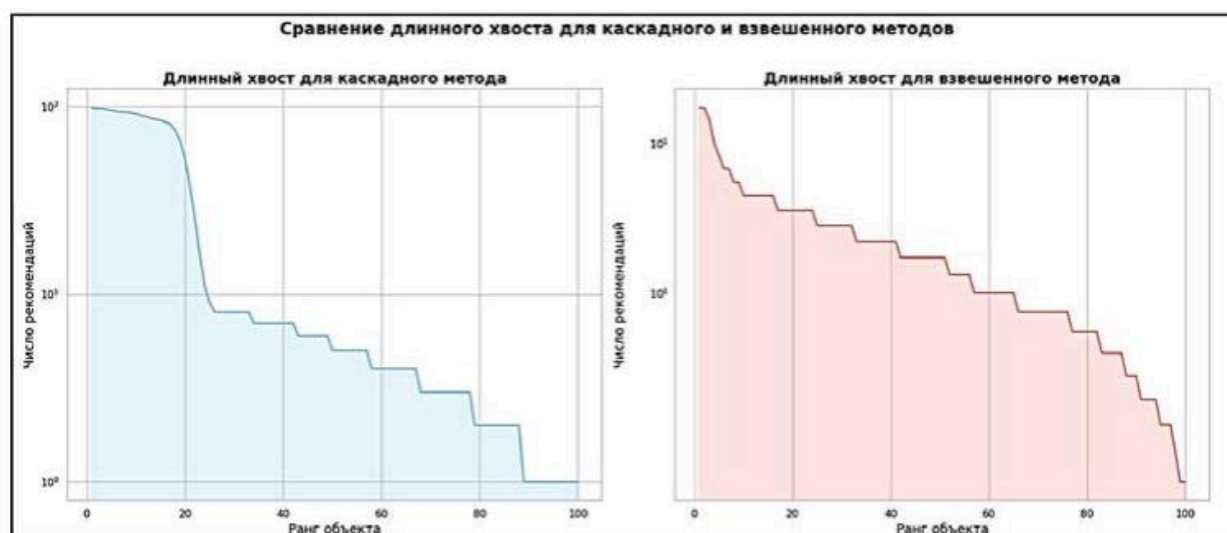


Рис. 4. Сравнение хвостов

Каскадный метод показывает более резкий спад с высокой концентрацией рекомендаций в «голове» списка, что указывает на ориентацию на популярные объекты. Это связано с тем, что каскадный метод, в силу своей структуры, тяготеет к выдаче часто рекомендуемых и популярных объектов, обеспечивая более узкий набор рекомендаций.

Взвешенный метод демонстрирует более плавное распределение по «длинному хвосту», охватывая более широкий диапазон объектов, включая менее популярные. Такой подход позволяет пользователям получать более разнообразные рекомендации, соответствующие их интересам, но не обязательно являющиеся самыми популярными.

Эти результаты подтверждают, что взвешенный метод лучше адаптируется к индивидуальным предпочтениям пользователей, предлагая более разнообразный контент, в то время как каскадный метод склонен поддерживать рекомендации, основанные на популярности, что может ограничить спектр предложений.

Графики демонстрируют, что взвешенный метод превосходит каскадный метод в аспектах персонализации. Данный фактор поможет улучшить пользовательский опыт за счет рекомендаций, более точно соответствующих индивидуальным предпочтениям. Анализ длинного хвоста также подчеркивает способность системы адаптироваться к широкой аудитории, предлагая, как популярные, так и редкие товары.

Обобщение и оценка результатов исследования. Результаты исследования показали, что наилучший метод объединения рекомендаций для системы — это взвешенный метод. Этот подход позволяет достичь высокой точности и релевантности рекомендаций, поскольку оптимально сочетает преимущества как коллаборативной, так и контентной фильтраций.

Важным преимуществом взвешенного метода является его гибкость: он предоставляет возможность внедрения дополнительных методов анализа, таких как учет отзывов и оценок пользователей, для улучшения качества рекомендаций. Благодаря этому пользователи смогут оставлять свои мнения о блюдах, что позволит системе лучше понять, какие аспекты важны для клиентов, и повышать точность рекомендаций на основе этих данных.

Литература

- [1] Пронин Д.С., Зарубин И.Б. Информационная рекомендательная система отбора и ранжирования произведений графического цифрового искусства с использованием комбинированного коллаборативно-контентного подхода. *ГрафиКон. Междунар. конф. по компьютерной графике и зрению: сб. тр.* Москва, Институт проблем управления им. В.А. Трапезникова РАН, 2023, с. 4–12.
- [2] Игнатьев В.Ю., Лемтюжникова Д.В., Руль Д.И., Рябов И.Л. Построение гибридной рекомендательной системы с улучшением точности. *Известия Российской академии наук. Теория и системы управления*, 2022, № 6, с. 3–7.
- [3] Козлова М.Г., Германчук М.С. Разработка гибридной системы рекомендаций. *Таврический вестник информатики и математики*, 2022, № 3 (56), с. 30–52.
- [4] Сенько О.В., Докукин А.А. Использование ансамблей с увеличенной дивергенцией в пространстве прогнозов в рекомендательных системах. *Автоматика и телемеханика*, 2021, № 3 (56), с. 2–7.
- [5] Куренных А.Е., Судаков В.А. Подход к разработке гибридных рекомендательных систем. *Бюллетень науки и практики*, 2022, т. 8, № 11, с. 378–382.
- [6] Ляликова В.Г., Безрядин М.М. Построение гибридной рекомендательной системы. *Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии*, 2021, № 4, с. 121–129.
- [7] Калугин К.Ю. Гибридная рекомендательная система в сфере малой электронной коммерции. *Прикладная математика и информатика: современные исследования в области естественных и технических наук. VII Междунар. науч.-практ. конф. (школа-семинар) молодых ученых: сб. материалов.* Тольятти, Тольяттинский государственный университет, 2021, с. 37–41.

Comparative analysis of methods for combining collaborative filtering and content - based filtering to build hybrid recommendation systems in an e-commerce web application

Gusko G.A.

ka-s-2015@mail.ru

Mikhailov A.S.

mikhailow.aleksei2018@yandex.ru

Vershinin E.V.

vershinin@bmstu.ru

BMSTU (KB), Kaluga, Russia

This research includes the development and analysis of an effective recommendation system implemented in a web application for ordering food, based on a hybrid approach that combines content and collaborative filtering. The advantages and disadvantages of various methods of combining recommendations, such as weighted combination, cascade, ensemble and multilevel approaches, are considered in detail. In this study, special attention is paid to choosing the optimal strategy that will improve the accuracy of recommendations.

Keywords: hybrid method, filtering, content filtering, collaborative filtering, combining methods in hybrid filtering