Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Кафедра информационно-аналитических систем

Смирнов Александр Львович

Построение гибридной рекомендательной системы новостей с применением методов оптимизации

Отчет по преддипломной практике

Научный руководитель:
к.ф.-м.н., доц. Михайлова Елена Георгиевна

Рецензент:
руководитель отдела инженерии Осипов Е.В.

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Department of Information and Analytical Systems

Alexander Smirnov

A Hybrid Approach for News Recommender System using Optimization Methods

Pre-Diploma practice report

Scientific supervisor:
professor Elena Mikhaylova

Reviewer:
head of engineering department Evgenii Osipov

Saint-Petersburg
2021

Оглавление

1. Введение

Рекомендательная система является важной частью каждого приложения, в котором содержится большое количество контента и действий пользователей. Огромный объем информации приводит к тому, что пользователь не может найти релевантный для себя контент.

Рекомендательные системы были испольюзуются для рекомендаций фильмов, музыки и книг (7) (9) (8). Рекомендательной системой называется любая система, которая выдает персонализированные рекомендации или направляет пользователя к интересным или полезным объектам в большом пространстве возможных вариантов. Такие системы имеют очевидное преимущество в среде, где объем информации значительно превосходит возможности человека ее исследовать.

Системы рекомендаций теперь являются неотъемлемой частью некоторых сайтов электронной коммерции, таких как Amazon.com и CDNow (12). Критерии «индивидуальности» и «интересности и полезности», отделяют рекомендательную систему от информационно-поисковых систем (1). Семантика поисковой системы - «соответствие»: система должна возвращать все те элементы, которые соответствуют запросу, ранжированные по степени релевантности. Такие методы, как обратная связь по релевантности, позволяют поисковой системе уточнить представление запроса пользователя и представляют собой простую форму рекомендации.

Сфера рекомендаций новостей имеет свою специфику: новости быстро стареют и это нужно учитывать.

Существует три основных типа рекомендаций: на основе памяти, на основе модели и гибридные (10). Методы на основе памяти (6) обычно используют метрики сходства для определения расстояния между двумя пользователями или двумя элементами. Методы, основанные на моделях, используют демографическую, информационную или агрегированную информацию для создания модели, которая генерирует рекомендации. Гибридные методы (3) комбинируют различные типы рекомендателей для повышения качества рекомендаций.

Общие подходы, такие как коллаборативная фильтрация, имеют свои проблемы: холодный старт, масштабируемость и разреженность данных. Контентные подходы страдают от того факта, что мы должны каким-то образом представлять рекомендуемый элемент в пространстве признаков.

В этой статье описана гибридная рекомендательная система.

Для единообразия в ходе работы мы перечисляем некоторые термины с их значениями:

- Рейтинг: скалярное значение, отражающее степень релевантности
  - явный (напрямую от пользователя, например, фильм с рейтингом пользователей)
  - неявный (выводится из активности пользователя, например, пользователь перестал смотреть фильм через 5 минут)
- Прогноз: оценка рейтинга
- Рекомендация: выбранные ранжированные элементы для пользователя
- Содержимое: атрибуты, текст и т. д .; все о рекомендуемом предмете

Остальная часть диплома организована следующим образом:

- 2 описывает связанные работы
- 3 содержит обзор нашего подхода
- 4 описывает входные данные
- 5 объясняет наш подход
- 6 предоставляет тесты и эксперименты, подтверждающие результаты нашей системы
- 7 представляет выводы

2. Связанные работы

Согласно исследованию (5), методы глубокого обучения не должны превосходить концептуально и вычислительно более простые алгоритмы, поэтому мы не будем их рассматривать.

Наша цель - выбрать оптимальный алгоритм для каждой из следующих задач, чтобы впоследствии комбинировать их:

- **Коллаборативная фильтрация**: создание прогнозов об интересах пользователя путем сбора информации о предпочтениях или вкусовых предпочтениях других пользователей. Он основан на предположении, что если человек A придерживается того же мнения, что и человек B, по какой-либо проблеме, A с большей вероятностью будет иметь мнение B по другому вопросу, чем мнение случайно выбранного человека; по этой теме было проведено много исследований, но статья (11) доказывает, что хорошо настроенный базовый подход SVD ++ превосходит недавно представленные алгоритмы глубокого обучения.

- **Фильтрация на основе содержимого**: основана на предположении, что людям, которым нравились новости с определенными содержимым в прошлом, понравятся такие же элементы и в будущем. Из истории собирается профиль предпочтений пользователя, который впоследствии сравнивается с кандидатами на рекомендации.

- **Фильтрация на основе текущей сессии**: рекомендует на основе активности пользователя в текущем сеансе.

- **Фильтрация по популярности**: использует такую информацию, как количество просмотров, показов, комментариев и т. д.

- **Демографическая фильтрация**: использует демографические данные, такие как возраст, пол, образование и т. д., для определения категорий пользователей.

- **Фильтрация на основе времени**: это способ, при котором более свежие элементы получают более высокие оценки

Также есть несколько способов (2) комбинировать рекомендатели между собой:

- **Взвешено**: оценки (или голоса) нескольких методов рекомендаций объединяются для получения единой рекомендации.
- **Переключаясь**: система переключается между методами рекомендаций в зависимости от текущей ситуации.
- **Смешанный**: рекомендации от нескольких разных рекомендателей отображаются одновременно.
- **Комбинированный**: функции из разных источников рекомендаций объединены в единый алгоритм рекомендаций.
- **Стекинг**: выходные данные одного метода используются как входные данные для другого.

Мы сталкиваемся с проблемой, что миллионы новостей теоретически подходят для рекомендации, поэтому использовать вышеупомянутые методы на таком большом массиве данных некорректно. Вместо этого, как указано в статье (4), мы хотим реализовать конвейер `генерация кандидатов` $\rightarrow$ `ранжирование`, чтобы уменьшить количество кандидатов.

3. Обзор решения

Нашей задачей является скомбинировать лучшие подходы в области рекомендательных систем. Решение состоит из двух частей:

- **Генерация кандидатов**: уменьшение количества рекомендуемых новостей. Эти кандидаты предназначены для фильтрации совсем не релевантного (напр., устаревшего) контента.
- **Ранжирование**: применяем лучшие в своей области алгоритмы для ранжирования оставшихся новостей

Архитектура представлена на Рис . 1:

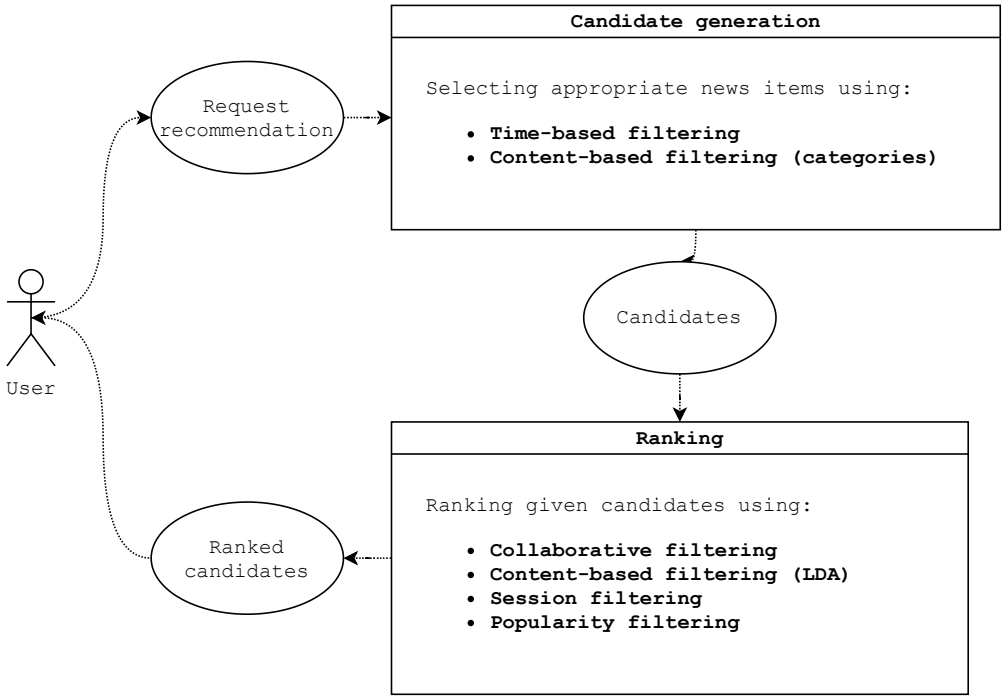


Рис. 1: архитектура

4. Входные данные

Поскольку мы решаем предметную задачу, у нас есть данные о предметной области. Наши данные состоят из информации о новостях и включают в себя следующие таблицы:

метаданные

Метаданные о новостях. Здесь у нас есть столбец *item_ id*, который обозначает уникальный идентификатор элемента, затем *date*, который показывает, когда этот элемент был выпущен. Столбец *source_ id* обозначает идентификатор издателя этой новости. Столбец *category* означает категорию текущей новости. Эта категория берется из текста новости по ключевым словам.

item_ id	*date*	*source_ id*	*category*
1	2021-01-08 22:08:39	9	politics, conflicts
2	2021-01-09 10:28:58	5	IT, social media
3	2021-01-09 14:20:34	12	accident
⋮	⋮	⋮	⋮

Таблица 1: метаданные

содержимое

В таблице содержимого содержатся тексты новостей в столбце *news_ content*. Это основная информация, которая используется нашей системой рекомендаций, потому что мы можем извлекать много ценных данных из текста, такие как темы.

item_ id	*news title*	*news content*
1	Azerbaijan denies reports on construction of Turkish air bases in the country	Information that Turkey will create air bases ...
2	Durov announced the massive transition of WhatsApp users to Telegram	Telegram developer Pavel Durov said in his channel ...
3	Passenger plane that disappeared from radar crashed	Passenger plane taking off from Jakarta, disappeared ...
⋮	⋮	⋮

Таблица 2: содержимое

показы & просмотры

Поскольку мы работаем с действиями пользователей, у нас должны быть журналы логов, например, какие элементы были нажаты и в какое время, поэтому у нас есть отдельная таблица, содержащая эту информацию. *shows* означает, что *item_id* был показан *user_id*, *views* - если *item_id* был нажат пользователем *user_id* .

user_id	*item_id*
10	1
10	2
23	1
23	3
23	2
38	3
38	1
⋮	⋮

Таблица 3: показы

user_id	*item_id*
10	1
10	2
23	1
38	3
⋮	⋮

Таблица 4: просмотры

эмоции & комментарии

Пользователь может оставлять явную обратную связь на новости, которые он смотрит. В частности, можно оставить emoji {😄, 😮, 😢, 😣, ♥ } или комментарий, которые мы анализируем впоследствие.

user_id	*item_id*	*emotion_id*
10	1	1
10	2	3
23	1	3
38	3	2
⋮	⋮	⋮

Таблица 5: эмоции

user_id	*item_id*	*comment*
10	1	that's great
10	1	wish it will continue
23	2	whatsapp is not competetive anymore
⋮	⋮	⋮

Таблица 6: комментарии

подписки на ленты

Каждая новость публикуется через какую-либо ленту, и у пользователя есть возможность подписаться на эту ленту. Таким образом, это может дать нам полезную информацию, если пользователь предпочитает контент из одной ленты контенту из другой ленты. Итак, следующая таблица показывает, подписался ли *user_id* на *feed_id*.

user_id	*source_id*
10	9
23	5
⋮	⋮

Таблица 7: факт подписки

5. Наш подход

Давайте подробно опишем конвейер ранжирования кандидатов.

Этап генерации кандидатов не только фильтрует элементы, но и присваивает им веса.

Обзор компонентов генерации кандидатов

Цель этапа генерации кандидатов - удалить, как правило, нерелевантные элементы, чтобы дополнительные алгоритмы не пострадали от объема данных. И фильтрация на основе времени, и фильтрация на основе содержимого имеют свой собственный вес. По мере того, как пользователь все больше взаимодействует с системой, фильтрация на основе содержимого увеличивает свой вес.

Используя рекомендации, описанные ниже, мы присваиваем баллы каждому элементу и выбираем самые популярные элементы стоимостью n (например, 50000).

5.0.1. Фильтрация по времени

Поскольку мы работаем с данными новостей, первый фильтр - это временной фильтр. Эта часть состоит из 2 шагов:

- **Фильтрация**: удалить все элементы старше 3-х дней
- **Рейтинг**: присвоить вес всем элементам, оставшимся после фильтрации

Для ранжирования мы будем использовать следующую формулу:

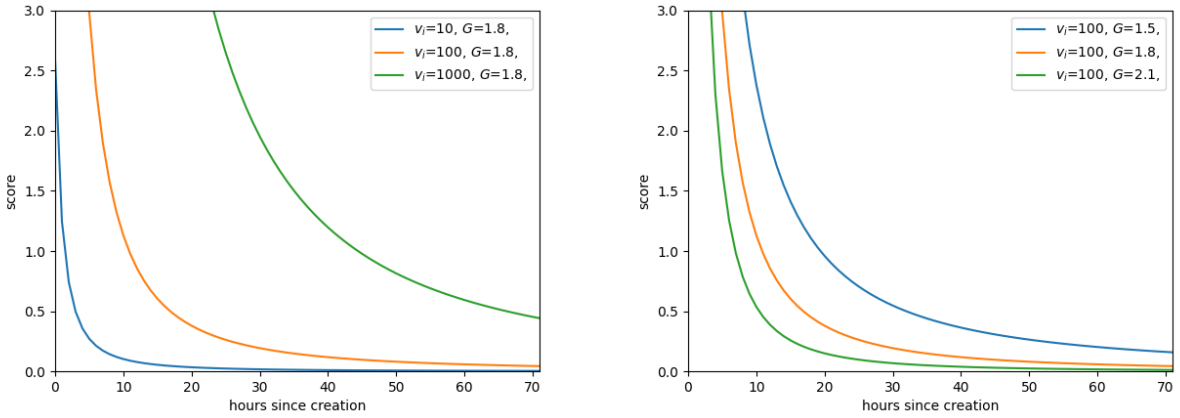
$$r_i = \frac{(v_i - 1)}{(t - t_i + 2)^G} \tag{1}$$

- r_i - счет за $item_i$
- v_i - количество просмотров $item_i$
- t - время прямо сейчас, t_i - время создания $item_i$, $t - t_i$ - часы, прошедшие с момента создания элемента

- G - фактор гравитации

Оценка уменьшается по мере увеличения $t - t_i$, что означает, что более старые элементы будут получать все более низкие оценки. v_i вычитается на -1, чтобы отрицать просмотр создателя новости. $t - t_i$ увеличивается на 2, поэтому, даже если $t - t_i = 0$, фактор силы тяжести G будет действовать. По умолчанию мы устанавливаем $G = 1.8$ и увеличиваем его, если хотим отдавать приоритет новым новостям над старыми и наоборот.

Ниже показаны зависимости между счетом и часами с момента создания:



(a) разное количество просмотров (b) разный фактор гравитации

Рис. 2: ранжирование по времени

В конце значиния нормализуются с помощью `min-max` нормализации.

5.0.2. Фильтрация на основе категорий

Мы представляем каждую новость как вектор значений достоверности (насколько сильно каждая новость относится к категории).

Тегирование происходит следующим образом: составляем словарь принадлежности слов к категориям:

Уверенность берется из метрики слова tf-idf вместе со всем набором документов.

item_id	*politics*	*IT*	*social_media*	$\cdots$	*confilcts*
1	0.55	0	0	$\cdots$	0.3
2	0	0.81	0.62	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Таблица 8: тексовые вектора

word	*category*
trump	politics
crash	accident
telegeram	IT
$\vdots$	$\vdots$

Таблица 9: категории слов

Количество слов ограничено, но в течение дня добавляются новые слова, а каждую ночь пересчитываются вектора элементов.

У нас есть действия, пока пользователь взаимодействует с системой, и веса для элементов для каждого взаимодействия (. 10).

action	*score*
shown	-1
viewed	2
emoji or comment	1
read till the end	1

Таблица 10: веса взаимодействий

Все данные агрегированы, и каждая новость имеет оценку действия в зависимости от того, что с ней сделал пользователь. Предположим, что пользователь просмотрел и оставил смайлик. Общий балл составит: -1 (пользователю был показан элемент) $+2$ (пользователь просмотрел элемент) $+1$ (пользователь оставил смайлик) $= 2$.

По мере взаимодействия пользователя с системой мы формируем его вектор предпочтений следующим образом:

$$u_i = \sum_{j=1}^{n} v_j \times s_j \tag{2}$$

- u_i – вектор $user_i$
- v_j – вектор $item_j$
- s_j – вес взаимодействия с $item_j$

Пользовательский вектор имеет такую же размерность, что и вектор элемента.

Поскольку теперь у нас есть векторы пользователей и элементов, мы можем найти между ними сходство:

$$r_{ui} = \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}} \tag{3}$$

- r_{ui} – рейтинг $user\ item$
- u – вектор $user$
- v – вектор $item$

Как было сказано ранее, фильтрация на основе содержимого имеет свой собственный вес воздействия, который вначале невелик (мы не хотим ограничивать доступ пользователя к контенту только потому, что он сделал несколько случайных щелчков), но он увеличивается по мере взаимодействия пользователя с системой и мы можем делать прогнозы относительно его категориальных предпочтений.

5.1. Обзор компонентов ранжирования

5.1.1. Коллаборативная фильтрация

Мы используем алгоритм SVD ++.

Для рекомендаций по коллаборативной фильтрации у нас должна быть матрица пользовательских оценок, которая формируется из истории оценок.

Мы используем модификацию Funk MF, которая факторизует матрицу рейтингов «пользователь-элемент» как произведение двух матриц меньшей размерности, первая имеет строку для каждого пользователя, а вторая имеет столбец для каждого элемента. Строка или столбец, связанные с конкретным пользователем или элементом, называются скрытыми признаками.

$$r_{ui} = \sum_{f=0}^{n} H_{u,f} W_{f,i} \tag{4}$$

Хотя Funk MF может обеспечить очень хорошее качество рекомендаций, но его способность использовать только явные числовые рейтинги в качестве взаимодействий между пользователями и элементами представляет собой ограничение. Современные рекомендательные системы должны использовать все доступные взаимодействия, как явные (например, числовые рейтинги), так и неявные (например, лайки, покупки, пропущенные, добавленные в закладки). С этой целью SVD ++ также был разработан с учетом неявных взаимодействий. По сравнению с Funk MF, SVD ++ также учитывает предвзятость пользователя и элемента.

Прогнозируемый рейтинг, который пользователь u поставит элементу i, вычисляется следующим образом:

$$r_{ui} = \mu + b_i + b_u + \sum_{f=0}^{n} H_{u,f} W_{f,i} \tag{5}$$

5.1.2. Фильтрация по популярности

Для измерения популярности мы аггрегируем следующие данные: показы, просмотры, emoji, комментарии.

Таким образом, когда мы применяем информацию о популярности новостных статей, мы можем выставлять им оценки по следующему алгоритму:

`min-max` нормализуем *shows_ num*, *views_ num*, *emotions_ num*, *comments_ n*
затем делим на 4 (чтобы получить 1 как максимум после суммы) и сум-

item_id	*shows_num*	*views_num*	*emotions_num*	*comments_num*
1	1043	231	52	7
2	828	478	78	11
3	163	25	5	0
⋮	⋮	⋮	⋮	⋮

Таблица 11: популярность элементов

мируем все эти значения.

$$r_i = \frac{shows\_num}{4} + \frac{views\_num}{4} + \frac{emotions\_num}{4} + \frac{comments\_num}{4} \quad (6)$$

5.1.3. Фильтрация на основе содежимого (LDA)

Для фильтрации на основе содержимого мы должны каким-то образом векторизовать новости и представлять предпочтения пользователей через эти векторизованные новости. Мы будем использовать латентное распределение Дирихле (LDA), которое представляет собой генеративную статистическую модель, которая позволяет объяснять наборы наблюдений группами, которые объясняют, почему некоторые части данных похожи. Например, если наблюдения представляют собой слова, собранные в тексты, предполагается, что каждый документ представляет собой смесь небольшого количества тем и что присутствие каждого слова связано с одной из тем документа.

Таким образом, мы можем векторизовать текст в зависимости от того, как каждый текст относится к каждой категории, от 0 до 1. Например, предположим, что у нас есть 3 темы, извлеченные с помощью модели LDA:

Допустим у нас текст: “Hackers use mobile emulators to steal millions of dollars”. Его векторизованной формой будет $[0.3, 0, 0.2]$.

Все вычисления проведены в соответствии с 5.0.2.

topic	*word*	*score*
1	dollar	0.3
1	bank	0.2
1	money	0.15
2	sugar	0.4
2	cooking	0.3
3	IT	0.25
3	hacker	0.2

Таблица 12: темы слов

5.1.4. Фильтрация на основе текущей сессии

Мы хотим мгновенно реагировать на действия пользователя, поэтому применяем фильтрацию на основе текущей сессии следующим образом: пытаемся найти новости, похожие на те, которые пользователь только что смотрел.

$$r_i = \sum_{k=0}^{n} similarity\{current\_item\_vector, last\_viewed\_vector_k\} \times weight_k \tag{7}$$

где $weight_k$ это вес последней просмотренной новости. Вес тем больше, чем недавнее была просмотрена новость.

5.2. Гибридная система

Таким образом, наличие всей этой информации позволяет нам настраивать вклад каждой отдельной системы с помощью поиска по сетке. Таким образом, мы оптимизируем веса, которые влияют на каждую рекомендацию. Конкретные оптимизируемые метрики будут указаны ниже.

6. Оценка качества

6.1. Обзор

Для оценки у нас есть информация о том, какой список рекомендаций был дан каждому пользователю и каков источник рекомендации:

user_id	*recommendation_list*	*content_based_filtering*	*collaborative_filtering*
2	{(2, 0.91), (1, 0.74), (3, 0.23)}	{(2, 0.45), (1, 0.54), (3, 0.08)}	{(2, 0.92), (1, 0.4), (3, 0.3)}
1	{(3, 0.73), (1, 0.69), (2, 0.15)}	{(2, 0.6), (1, 0.44), (3, 0.04)}	{(2, 0.58), (1, 0.58), (3, 0.14)}
⋮	⋮	⋮	⋮

Таблица 13: логи выдачи рекомендаций

- *recommendation_list*: список рекомендаций, который состоит из пар (*item_id*, *score*)
- *content_based_filtering & collaborative_filtering*: источники рекомендаций

Поскольку мы используем взвешенную сумму наших рекомендаций, у нас есть уникальные значения весов для каждого пользователя:

user_id	*content_based_filtering*	*collaborative_filtering*
1	0.25	1
2	1	0.5
⋮	⋮	⋮

Таблица 14: значения весов

Чтобы проиллюстрировать подсчет оценок рекомендаций, взгляните на 1-ю строку таблицы журналы рекомендаций. Мы видим рекомендацию $r = \{(2, 0.91), (1, 0.74), (3, 0.23)\}$ для пользователя $u = 2$, поэтому мы должны найти значения повышения для этого конкретного пользователя в таблице boost values: усиление контентной фильтрации b_{cbf} для u составляет 1, усиление совместной фильтрации b_{cf} составляет $0,5$. Итоговая оценка рассчитывается следующим образом: $0,91 = 0,45 * b_{cbf} + 0,92 * b_{cf} = 0,45 * 1 + 0,92 * 0,5 = 0,45 + 0,46$.

Также у нас есть информация о shows и views, поэтому мы можем отслеживать, какой элемент в списке рекомендаций был нажат, а какой элемент был пропущен, поэтому в зависимости от этого информация, которую мы можем отслеживать, понравился ли элемент пользователю или нет. Например, рассмотрим ситуацию, когда пользователь щелкнул 4-ю рекомендацию из списка рекомендаций. Это означает, что первые 3 рекомендации плохие, и мы должны штрафовать алгоритмы, которые отдавали приоритет этим элементам, учитывая тот факт, что 1-й элемент является худшим:

$$penalty = \frac{1}{num\_of\_item\_in\_rank} \tag{8}$$

7. Итоги

Система рекомендаций получила широкое распространение в разных сферах. Коллаборативная фильтрация фокусируется на рейтинге, игнорируя особенности самих элементов. Чтобы лучше оценивать предпочтения клиентов в отношении новостей, мы используем модель LDA для рассчёта предпочтения клиентов по новостным темам.

Чтобы прогнозировать рейтинг по новостям, мы принимаем во внимание схожесть клиентов и корреляцию между покупателями и новостями. Эксперимент показывает, что наш гибридный метод рекомендаций, основанный на характеристиках, лучше работает в нашем приложении для социальных сетей.

Мы предлагаем новый гибридный метод рекомендаций, основанный на функциях повышения качества выдачи.

Результаты показывают, что сочетание разных подходов приводит к увеличению вовлеченности пользователей. До внедрения рекомендательной системы пользователь тратил на вкладку новостей около 2 минут. Теперь, когда мы применили гибридную рекомендательную систему, пользователи проводят на вкладке новостей в среднем 5 минут.

Список литературы

[1] Belkin Nicholas, Croft W. Information Filtering and Information Retrieval: Two Sides of the Same Coin? // Commun. ACM. — 1992. — 12. — Vol. 35. — P. 29–38.

[2] Burke Robin. Hybrid Recommender Systems: Survey and Experiments // User Modeling and User-Adapted Interaction. — 2002. — 11. — Vol. 12.

[3] Combining Content-Based and Collaborative Filters in an Online Newspaper / M. Claypool, Anuja Gokhale, Tim Miranda et al. // SIGIR 1999. — 1999.

[4] Covington Paul, Adams Jay, Sargin Emre. Deep Neural Networks for YouTube Recommendations // Proceedings of the 10th ACM Conference on Recommender Systems. — New York, NY, USA, 2016.

[5] Dacrema Maurizio Ferrari, Cremonesi Paolo, Jannach Dietmar. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches // Proceedings of the 13th ACM Conference on Recommender Systems. — RecSys '19. — New York, NY, USA : Association for Computing Machinery, 2019. — P. 101–109.

[6] Delgado Joaquin, Ishii Naohiro. Memory-Based Weighted-Majority Prediction for Recommender Systems. — 1999. — 01.

[7] Duan L., Street W. N., Xu E. Healthcare information systems: data mining methods in the creation of a clinical recommender system // Enterprise Information Systems. — 2011. — Vol. 5, no. 2. — P. 169–181. — https://doi.org/10.1080/17517575.2010.541287.

[8] He Xu, Min Fan, Zhu William. A Comparative Study of Discretization Approaches for Granular Association Rule Mining // Canadian Conference on Electrical and Computer Engineering. — 2012. — 12. — Vol. 37.

[9] Min Fan, Zhu William. Mining top-k granular association rules for recommendation // Proceedings of the 2013 Joint IFSA World Congress and NAFIPS Annual Meeting, IFSA/NAFIPS 2013. — 2013. — 05.

[10] Recommender systems survey / J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez // Knowledge-Based Systems. — 2013. — Vol. 46. — P. 109–132. — URL: https://www.sciencedirect.com/science/article/pii/S0950705113001044.

[11] Rendle Steffen, Zhang Li, Koren Yehuda. On the Difficulty of Evaluating Baselines: A Study on Recommender Systems. — 2019. — 05.

[12] Schafer J. Ben, Konstan Joseph, Riedl John. Recommender Systems in E-Commerce // Proceedings of the 1st ACM Conference on Electronic Commerce. — EC '99. — New York, NY, USA : Association for Computing Machinery, 1999. — P. 158–166. — URL: https://doi.org/10.1145/336992.337035.