Ю. С. Нефедова, Архитектура гибридной рекомендательной системы GEFEST (Generation–Expansion–Filtering–Sorting–Truncation), *Системы и средства информ.*, 2012, том 22, выпуск 2, 176–196

АРХИТЕКТУРА ГИБРИДНОЙ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ GEFEST (GENERATION–EXPANSION–FILTERING–SORTING–TRUNCATION)*

Ю. С. Нефедова[1]

Аннотация: Рекомендательная система представляет сравнительно новый класс программного обеспечения, в задачу которого входит предсказание того, какие объекты (фильмы, музыка, книги, новости, веб-сайты) будут интересны пользователю, с помощью анализа его действий и оценок. В данной работе описано построение такой системы. Решается задача формирования подходящего для пользователя списка рекомендуемых товаров. Упор сделан на возможность использовать всю доступную информацию, содержащуюся в данных о товаре (описание, характеристика), данных о пользователе (профиль пользователя) и в данных о поведении пользователя (сведения об использовании или покупке товара и/или явная оценка товара).

Ключевые слова: рекомендательные системы; коллаборативная фильтрация; метод ближайших соседей; алгоритмы классификации

1 Введение

Рекомендательные системы — это программные средства, технологии и методы выработки полезных для пользователя предложений, советов и рекомендаций. Предоставленные предложения направлены на поддержку пользователя в различных процессах принятия решений, а именно: какие товары купить, какую музыку послушать или какие новости прочитать. Рекомендательные системы помогают пользователю ориентироваться в ситуации стремительно нарастающих объемов контента, они оказались ценным средством для борьбы с информационной перегрузкой и стали одним из самых мощных и популярных инструментов в электронной коммерции.

Рекомендательные системы в первую очередь предназначены для лиц, которые не имеют достаточного личного опыта или компетентности для оценки

*Работа поддержана Российским фондом фундаментальных исследований (проекты 11-01-12026-офи-м, 11-01-00515а), а также Министерством образования и науки РФ в рамках ФЦП «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы» (проект МК–2256.2012.1).

[1]Московский государственный университет им. М. В. Ломоносова, факультет вычислительной математики и кибернетики; Институт проблем информатики Российской академии наук, y.nefedova@gmail.com

подавляющего количества альтернативных объектов, которые веб-сайт или, например, интернет-магазин может предложить.

«Объект» является общим термином, используемым для обозначения того, что система рекомендует пользователям. Рекомендательные системы обычно фокусируются на определенном типе объекта, например: книгах, DVD-дисках, новостях, товарах.

В своей простейшей форме персональные рекомендации предоставляются в виде упорядоченных по рейтингу списков объектов. Для расчета рейтинга рекомендательные системы, основываясь на предпочтениях пользователя, пытаются предсказать наиболее подходящие товары или услуги. Для того чтобы реализовать такую вычислительную задачу, рекомендательные системы собирают от пользователей информацию об их предпочтениях, которые могут быть явно выражены, например как рейтинг товаров или оценка фильмов, или могут быть выведены путем интерпретации действий пользователя. Так, рекомендательная система может рассмотреть переход на определенную веб-страницу как неявный признак предпочтения для объектов, показанных на этой странице.

Рекомендательные системы являются относительно новым предметом исследования по сравнению с разработками многочисленных средств и методов в других классических информационных системах (например, в системах баз данных или поисковых системах). Рекомендательные системы выделилась в самостоятельную область исследований в середине 1990-х годов, однако мощнейшим толчком к развитию этой темы стал конкурс Netflix Prize [1], объявленный в 2006 г. компанией Netflix, одним из лидеров на американском рынке проката DVD. Компания предоставила участникам данные о 100 млн рейтингов, проставленных 480 тысячами пользователей 18 тысячам фильмов в течение 6 лет. Участник, показавший результаты на 10% лучше, чем их собственный движок рекомендаций, получал миллион долларов. Благодаря этому конкурсу начали бурно развиваться методы построения рекомендательных систем, однако задача оказалась сложной — несмотря на ажиотаж и жесточайшую конкуренцию, на выполнение условий у участников ушло три года.

В последние годы интерес к рекомендательным системам резко возрос, о чем свидетельствуют следующие факты.

1. Важную роль рекомендательных систем высоко оценили такие популярные интернет-сайты, как Amazon.com, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm, IMDb, Imhonet. Более того, многие медиа-компании в настоящее время разрабатывают и внедряют рекомендательные системы как часть услуг, которые они предоставляют своим абонентам.
2. Регулярно проводятся многочисленные конференции и семинары, относящиеся к данной области. В частности, специально основанная в 2007 г. ACM Recommender Systems (RecSys), по сей день являющаяся главным ежегодным событием в области исследований по рекомендательным технологиям. Кро-

ме того, сессии, посвященные рекомендательным системам, часто бывают включены в состав более традиционных конференций в области баз данных, информационных и адаптивных систем. Среди этих конференций стоит отметить ACM Special Interest Group on Information Retrieval (SIGIR), User Modeling, Adaptation and Personalization (UMAP) и ACM Special Interest Group on Management Of Data (SIGMOD).

3. В высших учебных заведениях по всему миру читаются специальные курсы, посвященные рекомендательным системам; учебники по рекомендательным системам являются крайне популярными; а также следует упомянуть недавно вышедшие книги [2, 3], вобравшие в себя уже накопившийся к этому времени немалый опыт построения рекомендательных систем.
4. Многие академические журналы издавали специальные выпуски, охватывающие исследования и разработки в области рекомендательных систем. Среди журналов, которые посвятили такие выпуски вопросам, связанным с рекомендательными системами, следует отметить: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer-Human Interaction (2005) и ACM Transactions on Information Systems (2004).

Развитие рекомендательных систем началось с довольно простого наблюдения: люди часто полагаются на рекомендации, предоставленные другими при принятии повседневных решений. Например, при выборе книги для чтения мы обычно рассчитываем на рекомендации друзей, работодатели учитывают рекомендательные письма при принятии решений о трудоустройстве, при выборе фильма для просмотра люди склонны ориентироваться на обзоры фильмов, которые написал кинокритик.

Стремясь подражать такому поведению, первые рекомендательные системы применяли алгоритмы, использующие рекомендации, выработанные сообществом пользователей, чтобы предоставить рекомендацию активному (прогнозируемому) пользователю. Эти рекомендации состояли из объектов, которые высоко оценили сходно мыслящие пользователи (с похожими вкусами). Такой подход стал называться *коллаборативной фильтрацией*. Его обоснование заключается в том, что если активный пользователь соглашался в прошлом с некоторыми пользователями, то новые рекомендации, поступающие от этих схожих пользователей, должны также представлять интерес для активного пользователя.

Современные же рекомендательные системы генерируют рекомендации, используя различные типы знаний и данных о пользователях, доступных объектах и предыдущих операциях пользователей над объектами. Затем пользователь может просматривать список рекомендаций. Он может принимать их или нет, может дать сразу или на следующем этапе явную или неявную реакцию, обеспечивая обратную связь. Все эти действия пользователя и обратная связь могут быть

сохранены в базе данных рекомендательной системы и могут быть использованы для создания новых рекомендаций при следующем взаимодействии пользователя и системы.

К настоящему времени можно выделить три основных подхода к построению рекомендательных систем:

1. **Collaborative filtering** (коллаборативная фильтрация). Как уже упоминалось выше, простая и оригинальная реализация этого подхода [4] рекомендует активным пользователям элементы, которые другим пользователям со схожими вкусами понравились в прошлом. Сходство во вкусе двух пользователей рассчитывается на основе сходства в рейтингах пользователей.

   Традиционный алгоритм коллаборативной фильтрации представляет пользователя как N-мерный вектор объектов, где N — число различных предлагаемых объектов (количество товаров в каталоге, фильмов, книг). Компоненты вектора положительны при покупке или явной положительной оценке пользователем соответствующего объекта и отрицательные для негативного по рейтингу объекта. Алгоритм генерирует рекомендации на основе рейтингов нескольких клиентов, которые наиболее близки к активному пользователю. Сходство двух пользователей A и B обычно измеряется косинусом угла между двумя пользовательскими векторами:

$$\text{similarity}\left(\overrightarrow{A}, \overrightarrow{B}\right) = \cos(\overrightarrow{A}, \overrightarrow{B}) = \frac{\overrightarrow{A} \bullet \overrightarrow{B}}{\|\overrightarrow{A}\| * \|\overrightarrow{B}\|}.$$

   Коллаборативная фильтрация считается самым популярным и широко применяемым подходом к построению рекомендательных систем.

2. **Content-based** [5]. Данный подход основан на использовании данных профилей пользователей и данных об объектах. Содержание профиля пользователя может состоять не только из истории его покупок и оценок объектов, но и из множества других параметров: возраста, места жительства и т. п. То же относится и к профилю (описанию) объектов. Данные в этом случае можно разделить по двум важным параметрам:

   (а) способ получения: явный (голосование, заполнение профиля и т. п.) или неявный (история покупок, географическое положение по IP-адресу, история посещений сайтов или же другая информация, собранная при помощи специализированных программ);

   (б) структурированность. Структурированными данными являются стандартные поля профиля, истории покупок, цена товара, тип объекта и т. п. К неструктурированным можно отнести свободные поля профиля (например, «о себе»), свободные комментарии об объектах, текстовые описания товаров, тексты новостей и т. п.

Основная сложность работы с неструктурированными данными — трудность «оцифровки» (например, приведение к векторному виду). Для работы с такими данными требуются специальные алгоритмы, например, нормализации слов и алгоритмы семантического анализа текстов (Latent Semantic Analysis, случайное индексирование [6] и т. п.). Обработка неструктурированных данных является очень сложной и не решенной до конца задачей, что говорит о перспективности данного направления, но в то же время и о необходимости крайне аккуратной работы с ними. Структурированные же данные довольно легко представляются в необходимом виде.

Следует отметить, что в классическом content-based подходе используются исключительно данные о товарах, причем рекомендации пользователю даются только на основе его личных покупок. Затем с помощью некой меры близости, часто — коэффициента Пирсона, определяются объекты, близкие к профилю пользователя.

3. **Гибридная рекомендательная система.** Различные подходы к созданию рекомендательных систем обладают своими достоинствами и недостатками, в связи с чем разумным представляется совмещение нескольких подходов в одной системе. Такие рекомендательные системы, основанные на комбинации описанных выше методов, называются **гибридными**. Гибридная система, сочетающая методы A и B, пытается использовать преимущества A, чтобы исправить недостатки B. Например, метод коллаборативной фильтрации страдает от проблемы «холодного старта», то есть этот метод не может рекомендовать объекты, которые не имеют рейтингов. Это не ограничивает content-based подход, у которого рекомендация новых объектов основана на их описании (характеристиках), что, как правило, легко доступно.

В заключение отметим, что разработка рекомендательных систем является многопрофильной задачей, требующей привлечения экспертов из различных областей, таких как искусственный интеллект, информационные технологии, интеллектуальный анализ данных, статистика, маркетинг.

2 Оценка качества работы системы

Важным пунктом для разработки рекомендательных систем является оценка качества работы таких систем. В этом разделе приводятся некоторые общепринятые метрики, такие как точность, полнота и т. п.

Следует отметить, что некоторые оценки хорошо применять лишь для систем, предсказывающих рейтинг, а некоторые — лишь для систем, дающих рекомендации типа Top N. Обычно область применения метрики видна из определения.

2.1 Точность (accuracy)

Существует множество различных метрик для оценки точности.

– **mean absolute error (MAE):**

$$\mathrm{MAE} = \frac{\sum\limits_{u \in U} \sum\limits_{i \in \mathrm{testset}_u} |\mathrm{rec}(u,i) - r_{u,i}|}{\sum\limits_{u \in U} |\mathrm{testset}_u|},$$

где u — пользователь; $i \in \mathrm{testset}_u$ — товар из тестовой выборки; $\mathrm{rec}(u,i)$ и $r_{u,i}$ — соответственно оцененный системой и реальный рейтинги товара i у пользователя u;

– **root mean square error (RMSE):**

$$\mathrm{RMSE} = \sqrt{\frac{\sum\limits_{u \in U} \sum\limits_{i \in \mathrm{testset}_u} (\mathrm{rec}(u,i) - r_{u,i})^2}{\sum\limits_{u \in U} |\mathrm{testset}_u|}};$$

– **normalized MAE and RMSE.** Значения MAE и RMSE делятся на $(r_{\max} - r_{\min})$ — разность между минимальным и максимальным рейтингом;

– **усредненные MAE и RMSE.** Если выборка неравномерна по товарам (пользователям), то можно рассчитывать MAE и RMSE для каждого товара (пользователя) отдельно, а затем брать среднее этих значений.

Метрики MAE и RMSE, как видно из определения, лучше работают с рейтингами (хотя унарный рейтинг (предлагать товар) тоже может быть использован).

В задачах, где рекомендательная система предлагает отсортированный список рекомендаций, можно использовать параметры, стандартные для различных систем классификации.

Можно выделить 4 типа исходов (ошибок или их отсутствия) для рекомендаций:

	Recommended	Not recommended
Used	true positive (tp)	false negative (fn)
Not Used	false positive (fp)	true negative (tn)

На множестве таких исходов можно построить следующие стандартные метрики:

– **точность (precision)**

$$P = \frac{\#\mathrm{tp}}{\#\mathrm{tp} + \#\mathrm{fp}};$$

– **полнота (recall)**

$$R = \frac{\#\mathrm{tp}}{\#\mathrm{tp} + \#\mathrm{fn}};$$

– **уровень ложноположительных (fp) ответов**

$$\mathrm{FP} = \frac{\#\mathrm{fp}}{\#\mathrm{fp} + \#\mathrm{tn}};$$

– **аккуратность (accuracy)**

$$A = \frac{\#\mathrm{tp} + \#\mathrm{tn}}{\#\mathrm{tp} + \#\mathrm{tn} + \#\mathrm{fp} + \#\mathrm{fn}}.$$

Значения этих метрик, очевидно, зависят от длины списка выдаваемых рекомендаций. Например, при увеличении длины списка полнота увеличится, а точность упадет. В связи с этим можно в качестве оценок использовать не только сами значения параметров, но и их отношения при изменении длины списка, так называемые Recieving Operating Characteristic: например, график соотношения P к R или R к FP.

Также существуют общепринятые объединенные метрики:

$$F_1 = \frac{2PR}{P+R};$$

$$F_\beta = (1+\beta^2)\frac{2PR}{\beta^2 P + R}.$$

Часто используются параметры $\beta = 0{,}5$ и 2.

2.2 Метрики ранжирования

Если пользователю рекомендуется несколько товаров, важно учитывать, в каком порядке данные товары были рекомендованы. Можно реализовать множество таких метрик. Приведем пример одной из них.

Пусть hits_u — множество товаров, правильно рекомендованных пользователю u. Тогда в качестве метрики для оценки ранжирования можно взять, например,

$$\mathrm{rankscore}_u = \sum_{i \in \mathrm{hits}_u} \frac{1}{2^{(\mathrm{rank}(i)-1)/\alpha}},$$

где α — параметр, $\mathrm{rank}\,(i)$ — каким номером был выдан товар в рекомендации.

Можно нормализовать такую оценку разделив на

$$\mathrm{rankscore}_u^{\max} = \sum_{i \in \mathrm{testset}_u} \frac{1}{2^{(\mathrm{idx}(i)-1)/\alpha}}\,,$$

где $\mathrm{idx}\,(i)$ — функция перечисления всех товаров из тестовой выборки для пользователя u.

2.3 Покрытие

Рассмотрим метрики покрытия на примере товаров. Для пользователей такие метрики могут быть введены аналогично.

Возможная выдача: #(Items_which_can_be_recommended)/#(All_items).

Также для оценки равномерности покрытия можно учитывать вероятность $p(i)$ рекомендации (или выбора) товара i. В качестве метрик можно взять:

- **Gini Index** $1/(n_{\mathrm{items}} - 1) \sum\limits_{j=1}^{n_{\mathrm{items}}} (2j - n_{\mathrm{items}} - 1) p(i_j)\,,$ последовательность номеров товаров i_j упорядочена по неубыванию $p(i_j)$;
- **энтропию Шеннона** $\sum\limits_{i=1}^{n_{\mathrm{items}}} p(i) \log p(i)\,.$

2.4 Диверсифицированность выдачи

Численно можно оценить, насколько диверсифицирована выдача рекомендаций для пользователя:

$$\mathrm{ILS}_u = \sum_{i \in \mathrm{recset}_u} \sum_{j \in \mathrm{recset}_u, i > j} \mathrm{sim}\,(i,j)\ ,$$

где $\mathrm{sim}\,(i,j)$ — некоторая функция, показывающая близость товаров i и j.

3 Постановка задачи

В данной работе описано построение архитектуры гибридной рекомендательной системы для решения следующей задачи: для пользователя необходимо выдавать список заданного числа наиболее подходящих к рекомендации объектов (например, товаров из интернет-магазина, книг, фильмов, мобильных приложений). При этом для каждой рекомендации желательно получать вещественный

параметр, характеризующий важность этой рекомендации. Также крайне желательно по каждой рекомендации пользователю давать объяснение, почему эта рекомендация была дана (например, функция «why?» на Amazon.com).

Основная важная особенность поставленной задачи — это необходимость использовать сразу все доступные данные:

- данные профиля пользователя (пол, возраст, местоположение, если есть возможность определить по GPS, интернет-активность, частота и длительность использования тех или иных мобильных приложений и т. п.);
- данные о товаре (описание и характеристики);
- данные покупок (рейтингов) — либо явные оценки пользователя, либо собранная информация об использовании или покупке товаров.

4 Архитектура гибрида. Общая идея

В ходе разработки архитектуры рекомендательной системы удалось совместить различные идеи в один общий гибрид. Данный гибрид имеет несколько этапов работы, в связи с чем на каждом этапе можно использовать подходы, методы и алгоритмы, наиболее подходящие для решения поставленной задачи.

4.1 Основные этапы

Предложенная архитектура системы представляет собой шестиступенчатый гибрид, который было предложено назвать GEFEST в соответствии с его основными этапами:

- **G**eneration — генерация начального списка возможных рекомендаций;
- **E**xpansion — расширение этого списка схожими товарами;
- **F**iltering — фильтрация расширенного списка;
- **E**stimation — оценка, прогнозирование рейтингов для товаров;
- **S**orting — сортировка по рейтингу и/или вероятности рейтинга (точности прогноза);
- **T**runcation — усечение, укорачивание отсортированного списка.

Данное разделение на этапы видится очень удачным по следующим причинам:

1. На первом этапе формируется лишь список рекомендаций, соответственно, не требуется от рекомендательной системы выставления рейтингов. Таким образом, могут использоваться системы типа Top N, которые сложно использовать в общем случае. Кроме того, отпадает необходимость строго следить за числом выдаваемых рекомендаций: верхняя граница может быть очень высока и выбирается лишь за счет вычислительной сложности системы.

2. Хорошее прогнозирование рейтингов, особенно если необходимо еще и оценивать достоверность прогноза, является самой вычислительно трудной задачей, т. к. могут быть использованы сложные алгоритмы и классификаторы. Часто используются отдельные классификаторы для каждого товара. Соответственно, прогнозировать рейтинг желательно для как можно меньшего числа товаров. В простых подходах к построению рекомендательных систем приходится для формирования рекомендации вычислять рейтинг для каждого товара, что с учетом большого и постоянно увеличивающегося количества товаров в решаемой задаче недопустимо. Наличие этапов 1–3, не требующих прогнозирования рейтинга, решает эту проблему.

3. Для каждого этапа можно вводить свои метрики оценки качества.

4.2 Требования к данным

Рассматриваемая ниже реализация предложенной идеи требует наличия определенных данных.

Во-первых, это, конечно же, **матрица рейтингов** $R = \{r_{ij}\}$ размера $n_{\text{users}} \times \times n_{\text{items}}$, где n_{users} — число пользователей, n_{items} — количество объектов-товаров, r_{ij} — вещественнозначный рейтинг, оценка, выставленная явным способом пользователем i товару j либо подсчитанная путем учета частоты и длительности использования данного приложения, покупки или возврата товара в интернет-магазине, посещение веб-страниц с описанием товара и т. п.

Этап 1 (Generation) требует **функции сравнения профилей пользователей** $\text{sim}_{\text{user}}(i, j)$, где i, j — номера пользователей.

Этап 2 (Expansion) требует **функции сравнения профилей товаров** $\text{sim}_{\text{item}}(i, j)$.

На этапе 4 (Estimation) уже требуются векторные представления профилей пользователей и товаров. В этом снова наблюдается преимущество предлагаемого подхода: некоторые неструктурированные данные легко сравнивать, но сложно кодировать в виде векторов. Такие данные можно использовать на этапах 1 и 2, а на этапе 4 не использовать.

Рассмотрим работу каждого этапа более подробно.

5 Generation. Формирование начального списка возможных рекомендаций

На данном этапе должен сформироваться как можно больший список возможных рекомендаций. Планируется использовать данные рейтингов и профилей пользователей.

Результатом работы данного этапа должно стать множество товаров $I_1 = i_1, i_2, \ldots, i_n$, которые хотя бы потенциально могут быть интересны пользователю.

Для обработки данных матрицы рейтингов R планируется использовать item-based collaborative filtering (CF) — модификацию классического подхода к коллаборативной фильтрации, предложенную компанией Amazon [7]. В данном методе, в отличие от классической коллаборативной фильтрации, сравниваются не пользователи, а товары. Метрика и составление векторов товаров могут быть использованы те же, что и в случае сравнения пользователей.

5.1 Item-based collaborative filtering

Первый шаг — формирование списка $I_{i-b\mathrm{CF}}$ при помощи Item-based collaborative filtering. Особенность его использования в данном случае — очень высокий порог схожести товаров, которые надо оставлять в рекомендации. В общем-то, этот порог можно сделать неограниченным (т. е. оставлять все товары, хоть как-то схожие по матрице R). Item-based CF, пожалуй, идеально подходит для этого этапа. Причины этого в следующем:

- простота;
- большинство вычислений происходит в фоновом режиме;
- очень высокая скорость работы в реальном времени;
- используются все данные матрицы R;
- формируются неожиданные рекомендации;
- частично решается проблема холодного старта по пользователям;
- используется разреженность данных;
- метод отлично проверен и, пожалуй, является одним из лучших среди CF.

Недостатки метода: холодный старт по товарам и то, что не используются данные о товарах и пользователях, — будут решены другими частями гибрида.

Итак, после работы метода получается список $I_{i-b\mathrm{CF}}$ потенциально рекомендованных товаров на основе данных матрицы R.

5.2 Ближайшие соседи по профилям пользователей

Здесь формируется список товаров $I_{\mathrm{up}NN}$. Пусть требуется выработать рекомендацию для пользователя под номером i. Тогда среди всех остальных пользователей выбирается n (параметр алгоритма) таких, у которых функция $\mathrm{sim}_{\mathrm{user}}(i, j)$ максимальна. Вычисление $\mathrm{sim}_{\mathrm{user}}(i, j)$ для всех пользователей довольно ресурсоемко, в связи с чем предлагается вынесение части вычислений в фоновый режим. А именно будем хранить для каждого пользователя i список n ближайших к нему

пользователей и соответствующие значения функции $\mathrm{sim}_{\mathrm{user}}(i, j)$. Обновление списков можно делать следующим образом:

- с некоторой регулярностью обновлять все списки;
- при добавлении нового пользователя или изменении профиля пользователя. В данном случае, пожалуй, разумно обновлять только список нового или измененного пользователя.

Далее для n ближайших пользователей формируется множество всех товаров $I_{\mathrm{up}NN}$, которые эти пользователи положительно оценили.

Плюсы:

- простота;
- возможность выносить вычисления в фоновый режим;
- очень высокая скорость работы в реальном времени;
- используются данные профилей пользователей;
- в паре с Item-based CF максимально решается проблема холодного старта для пользователей;
- не обязательно наличие векторной модели профиля пользователя;
- можно не особо аккуратно подбирать параметр n и брать его достаточно большим.

Минусы метода: холодный старт по товарам и то, что не используются данные о товарах, — устраняются другими частями гибрида.

5.3 Добавление товаров пользователя

Важно, что к множеству I_1 необходимо добавить и все товары, положительно оцененные пользователем — список I_{user}.

5.4 Формирование общего множества рекомендаций

Множество I_1 формируется объединением $I_{i-b\mathrm{CF}} \cup I_{\mathrm{up}NN} \cup I_u$.

6 Expansion. Расширение списка возможных рекомендаций

Так как данные рейтингов и профилей пользователей уже задействованы, расширение будет проводиться на основе данных профилей товаров. Будет использоваться метод ближайших соседей по профилям товаров. Для каждого товара $i \in I_1$ формируется множество Sim_i из n (параметр алгоритма) товаров с наиболее большим значением $\mathrm{sim}_{\mathrm{item}}(i, j)$ (сам товар i тоже принадлежит этому множеству, т. е. $i \in \mathrm{Sim}_i$). Как и в случае с пользователями, вычисление

$\mathrm{sim}_{\mathrm{item}}(i, j)$ для всех товаров довольно ресурсоемко и его можно также вынести в фоновый режим. А именно будем хранить для каждого товара i список n ближайших к нему товаров и соответствующие значения $\mathrm{sim}_{\mathrm{item}}(i, j)$. Обновление списков можно делать следующим образом:

– с некоторой регулярностью обновлять все списки;

– при добавлении нового товара или изменении описания какого-либо товара.

Итогом работы этапа будет множество товаров $I_2 = \bigcup\limits_{i \in I_1} \mathrm{Sim}_i$.

Плюсы:

– простота;

– возможность вынесения вычислений в фоновый режим;

– вкупе с первым этапом максимально решается проблема холодного старта для товаров;

– используются данные профиля;

– можно не особо аккуратно подбирать параметр n и брать его достаточно большим, не оптимизируя его значение на этом этапе.

Минус — не используются данные рейтингов и профилей пользователей. Эта проблема решается другими частями гибрида.

7 Общие выводы для этапов 1–2

7.1 Плюсы и минусы

\+ Все используемые методы просты.

\+ Решается проблема холодного старта как для пользователей, так и для товаров.

\+ Используются все возможные данные, кроме контекстных.

\+ Все методы масштабируемы по пользователям и товарам.

\+ Большинство вычислений могут быть вынесены в фон.

\+ Не обязательны векторные модели профилей пользователей и товаров.

\+ Список I_2 может быть на порядки меньше n_{items}, что существенно упрощает работу следующих этапов.

\+ Возможность объяснения рекомендаций (см. ниже).

– Большинство товаров в списке I_2 являются плохими рекомендациями (проблема решается на последующих этапах).

7.2 Оценка качества работы

Для I_2 основной и, пожалуй, единственной важной метрикой будет полнота. Пусть I_i — все товары, положительно оцененные пользователем i. Разделим их на обучающие I_i^{train} и тестовые I_i^{test}. Тестовые рейтинги будем считать нулевыми. После этого проведем шаги 1–2 и получим I_2^{test}. Необходимо добиться максимального значения $|I_2^{\text{test}} \cap I_i|/|I_i|$.

7.3 Объяснение рекомендаций

Существенным плюсом использования метода ближайших соседей является возможность объяснения рекомендаций. Если товар из множества I_2 в итоге попадает в выдачу рекомендаций, причину его появления можно легко объяснить пользователю исходя из того, как он попал в список I_2. Объяснения могут быть следующих типов:

- товар похож на товар i, купленный Вами;
- товар похож на товар i, купленный пользователем u, похожим на Вас;
- товар купил (оценил) пользователь u, похожий на Вас;
- товар покупают вместе с товаром i, купленным Вами, и т. д.

8 Filtering. Фильтрация списка рекомендаций

На данном этапе происходит фильтрация множества I_2. Из I_2 должны быть исключены заведомо неподходящие рекомендации. На выходе должно получиться $I_3 \subseteq I_2$, причем $I_2 \setminus I_3$ — заведомо плохие рекомендации.

8.1 Простейшая фильтрация

Убираем из множества I_2 товары I_u, уже купленные пользователем. Также можно убрать дубликаты товаров (если имеется возможность их поиска).

8.2 Knowledge-based фильтрация

Knowledge-based системы рекомендуют объект, основываясь на специальных знаниях о том, какие определенные функции объекта отвечают потребностям пользователя и его предпочтениям, и, в конечном счете, насколько объект полезен для пользователя. Их разработка является трудоемкой деятельностью ввиду необходимости создания правил. В связи с этим, не останавливаясь подробно, обозначим лишь некоторые факты.

Желательно реализовать две системы Knowledge-based фильтрации:

(1) **интерактивную**, когда пользователь сам указывает, что он ищет. Например, тип товара (электротехника, игры, художественные фильмы и т. п.), ограничения по стоимости товара и т. п.;

(2) **неинтерактивную**, где фильтрация происходит по параметрам пользователя.

8.3 Context Aware фильтрация

Context Aware фильтрация является усовершенствованием Knowledge-based фильтрации и вводит в правила и данные Knowledge-based еще и контекст. Например: время, когда был куплен товар, местоположение пользователя, когда была совершена покупка и т. п. Разработка такой фильтрации еще более сложна, нежели КВ, поэтому, пожалуй, нецелесообразна на начальном этапе.

Плюсы и минусы:

\+ убираются заведомо неправильные рекомендации;

\+ возможность использования разных фильтров;

\+ увеличена скорость работы фильтрации за счет этапов 1–2;

\+ присутствие фильтрации до прогнозирования рейтингов, а не после формирования списка рекомендаций позволяет еще больше упростить работу классификаторам, задействованным в оценке рейтинга;

– очень высока сложность реализации систем Knowledge-based.

9 Estimation. Прогнозирование рейтингов

Данный этап, пожалуй, является наиболее интересным и дает возможность использовать наиболее современные методы обработки данных и прогнозирования. На вход на данном шаге подается множество $I_3 = \{i_1, i_2, \ldots, i_n\}$, на выходе должно получиться множество пар товар–рейтинг: $\mathrm{RI} = \{(i_1, r_1), (i_2, r_2), \ldots, (i_{|I_3|}, r_{|I_3|})\}$. Или, что более предпочтительно, для каждого товара $i \in I_3$ получить список (если, например, рейтингов пять) $(p_1, p_2, p_3, p_4, p_5)$, где p_i — прогноз вероятности того, что рейтинг равен i.

Стоит снова упомянуть, что важнейшим плюсом предлагаемой системы является то, что множество I_3 относительно небольшое, к тому же его размер можно регулировать параметрами предыдущих шагов. В связи с этим для прогнозирования рейтингов можно использовать достаточно сложные системы, что не скажется критически на времени формирования рекомендации.

Главной сложностью при разработке этапа Estimation является необходимость использовать все три типа данных: профили пользователей, профили товаров и

рейтинги. Как уже отмечалось ранее, профили пользователей и описания товаров должны быть представлены в векторном виде.

Введем обозначения:

$R = \{r_{ij}\}$ — как и раньше, матрица рейтингов;

u_i — вектор рейтингов пользователя $u_i = (r_{i1}, r_{i2}, \ldots, r_{in_{\text{items}}})$;

a_i — вектор рейтингов товара $a_i = (r_{1i}, r_{2i}, \ldots, r_{n_{\text{users}}i})$;

pr_i — вектор профиля i-го пользователя $\text{pr}_i = (\text{pr}_{i1}, \text{pr}_{i2}, \ldots, \text{pr}_{in_{\text{pr}}})$, где n_{pr} — длина вектора профиля пользователя;

d_i — вектор описания i-го товара $d_i = (d_{i1}, d_{i2}, \ldots, d_{in_d})$, где n_d — длина вектора описания товара.

9.1 Выбор классификатора

Для хорошего прогнозирования необходим хороший современный классификатор, обладающий следующими свойствами:

- быстрая скорость классификации (как минимум необходимо наличие обучения);
- высокое качество прогнозирования;
- возможность оценки достоверности прогнозирования;
- возможность легкого дообучения (не обязательно, но крайне желательно).

И если метод ближайших соседей отлично подходил для формирования предварительного списка, то он уже не подходит ни по одному из вышеперечисленных пунктов, кроме последнего.

9.2 Проблемы применения классификаторов на используемых данных

Допустим в системе имеется какой-либо классификатор, на вход которого подается информация о пользователях. Вся информация о пользователе i — это векторы u_i и p_i. Логично записать компоненты этих векторов в один вектор («склейка» векторов) $(\text{pr}_i, u_i) = (\text{pr}_{i1}, \text{pr}_{i2}, \ldots, \text{pr}_{in_{\text{pr}}}, r_{i1}, r_{i2}, \ldots, r_{in_d})$ и подавать его на классификатор. В таком подходе возникают следующие проблемы:

Разреженность и большая размерность. Получившийся вектор будет очень сильно разрежен, а далеко не все классификаторы могут это использовать. Соответственно, за счет огромного числа товаров не использующие разреженность классификаторы будут работать непозволительно для рекомендательной системы медленно.

Нефиксированная размерность (масштабируемость). Практически все (а возможно, и все) классификаторы с обучением требуют фиксированной размерности подаваемых векторов. Длина вектора профиля пользователя будет меняться крайне редко, а вот длина u_i — очень часто. Получается, что при добавлении в систему нового товара необходимо полностью перестраивать классификаторы, что непозволительно.

Несбалансированность. Части u_i и pr_i в общем и целом должны иметь одинаковый или, по крайней мере, регулируемый приоритет. Однако в итоговом векторе полностью отсутствует баланс: часть u_i на порядки длиннее pr_i, pr_i — полностью заполненный вектор, а u_i — очень сильно разреженный. Для некоторых классификаторов такой дисбаланс может критически сказываться на качестве работы.

Следует отметить, что данные проблемы являются чисто техническими, а не относящимися к качеству рекомендательной системы. Для их решения предлагается использовать один простой и очень хорошо апробированный метод.

9.3 Снижение размерности случайным кодированием

Рассмотрим простейший случай бинарных векторов $B^n = 0, 1^n$, где n очень большое. Пусть также на этих векторах есть какая-то метрика близости (x, y), например $1 - cos(x, y)$. Имеется некоторое множество таких векторов $M \subset B^n$, причем векторы в этом множестве разрежены. Задача состоит в следующем: необходимо задать некоторое отображение $f : B^n \to B^m$, где $m < n$, так, чтобы для векторов из норма примерно сохранилась, т. е. $E(x, y) > E(u, v) \Rightarrow$ $\Rightarrow E(f(y), f(z)) > E(f(u), f(v))$ для большинства векторов x, y, u, v из M. Существует хороший и апробированный метод, позволяющий это сделать. Данный метод показал отличные результаты в обработке текстов, причем такое снижение размерности не только ускоряло, но и улучшало качество работы классификаторов, видимо, за счет добавления некоторой небольшой «нечеткости».

Сопоставим каждому числу $i = \overline{1, n}$ случайный разреженный вектор $y(i) \in$ $\in B^m$ с k единицами и $(m - k)$ нулями. Тогда для $x \in B^n$

$$f(x) = \bigvee_{i:x_i=1} y(i) = \bigvee_i x_i \cdot y(i)\,,$$

т. е. для векторов $y(i)$, соответствующих единицам в векторе x, делается побитовое OR.

Параметрами такого алгоритма являются m и k, причем m надо взять как можно меньшим. Эти параметры элементарно подбираются по множеству M и тесту $E(x, y) > E(u, v) \Rightarrow E(f(y), f(z)) > E(f(u), f(v))$.

9.3.1 Обобщение на целочисленные векторы

На целочисленные векторы метод можно обобщить естественным образом: формула

$$f(x) = \bigvee_i x_i \cdot y(i)$$

остается неизменной, а вместо $\bigvee$ используется покомпонентный максимум. Конечно же, m в таком случае заметно увеличится.

9.3.2 Пример использования в рекомендательных системах

Рассмотрим использование случайного кодирования на примере, описанном в пункте 9.2 Будем уменьшать размерность вектора u_i. Т. е. каждому товару при его появлении в базе будет приписываться случайный разреженный бинарный вектор размерности m, где m заметно меньше n_{items}. Теперь вектор u_i будет формироваться «сложением» (в смысле покомпонентного максимума) товаров, рейтинги которых для пользователя нам известны. То же самое можно сделать и для вектора a_i.

Такой подход решает **все** перечисленные в подразд. 9.2 проблемы, кроме проблемы сбалансированности, так как полученный вектор u_i все равно будет разреженным в большинстве случаев, а pr_i — нет. Для многих классификаторов это не страшно, однако для наиболее быстрых (например, линейного или нейронных сетей с малым числом слоев) неразреженные данные вообще плохо подходят. Для них необходима некоторая «средняя» разреженность (чтобы классы могли разделиться, но чтобы размерность оставалась в удобных для работы пределах). Векторы u_i можно сделать очень хорошо подходящими для таких классификаторов. Векторы pr_i также можно превратить в такие же векторы ровно таким же способом расширения размерности, вся разница лишь в том, что m в данном случае больше n. Такой метод используется в зрительной системе человека и отлично проявил себя, например, в задачах распознавания образов.

Кроме того, что очень важно и отличает данный метод от других методов снижения размерности, при новой оценке пользователя все, что необходимо сделать, — это прибавить к его уже сформированному вектору u_i соответствующий вектор товара, умноженный на рейтинг.

9.4 Прогнозирование рейтингов

Планируется использовать один большой классификатор, имеющий n_{ratings} классов, где n_{ratings} — количество возможных рейтингов, т. е. классами будут $r = 1, r = 2, \ldots, r = n_{\text{ratings}}$. На вход классификатора будет подаваться вся информация о паре (рейтинг, товар). Например, если пользователь i поставил

товару j оценку «4», то на обучение, как элемент класса $r = 4$, подается вектор $(\mathrm{pr}_i, u_i, a_j, d_j)$, где векторы u_i и a_j получены методом снижения размерностей. На обучение подаются все известные рейтинги.

Соответственно, для получения предполагаемого рейтинга на классификатор подаются четыре «склеенных» вектора $(\mathrm{pr}_i, u_i, a_j, d_j)$.

9.5 Возможность использования Context Aware

Именно на этом этапе можно наиболее активно использовать контекст, например, подавать на классификатор еще и вектор контекстных данных (время, местоположение и т. п.), т. е. в вектор, попадающий на обучение или на классификацию, добавится вектор контекста, например:

$$c(\mathrm{year}, \mathrm{season}, \mathrm{month}, \mathrm{day}, \mathrm{day\_or\_night}, \mathrm{hour}, \mathrm{country}, \mathrm{city}, \mathrm{network\_type}, \ldots) .$$

Тогда вектор, подаваемый на классификатор для обучения или классификации, будет иметь вид $(\mathrm{pr}_i, u_i, a_j, d_j, c)$. Соответственно, необходимо хранить c для каждой покупки и/или использования. Для вектора c может быть актуально расширение размерности.

В данном случае можно использовать встроенные возможности OLAP-систем (в общем-то, те же классификаторы). Реализация CA и уж тем более эффективная работа с OLAP-системами является непростой задачей, однако включение какого-либо элементарного и очевидного контекста, несомненно, повысит точность рекомендаций.

9.6 Взвешенный гибрид и адаптация

В связи с наличием огромного числа различных классификаторов, разумно на этапе 4 использовать их взвешенный гибрид.

Пусть имеется n_c классификаторов. Для пользователя i и товара j классификатором k были предсказаны вероятности рейтинга $p_1^k, p_2^k, \ldots$ Тогда итоговыми вероятностями будут

$$p_r^k = \frac{\sum\limits_{k=1}^{n_c} w_k p_r^k}{n_c} ,$$

где w_k — веса классификаторов, изменяемые адаптацией.

Кроме того, именно в классификаторах четвертого этапа скрыто большинство управляемых параметров, по которым может производиться адаптация системы.

10 Sorting и Truncation. Формирование окончательной рекомендации

На данном этапе формируется окончательная рекомендация. Простейший способ — отсортировать все товары из I_3 по рейтингу с максимальной вероятностью или среднему значению рейтинга $\sum p_i i$ и выдать N лучших. Кроме того, можно разбить выдачу на категории или интерактивно просить пользователя задать число выводимых рекомендаций.

11 Общие замечания

11.1 Плюсы и минусы

+ Разбиение на этапы имеет простую и естественную структуру.
+ Используются **все возможные данные** как для формирования начального списка, так и при формировании рейтингов.
+ На этапах 1–2 используются простые и апробированные методы.
+ Небольшое относительно n_{items} число элементов, попадающих на классификатор, позволяет использовать сложные современные классификаторы.
+ Решена проблема холодного старта как для товаров, так и для пользователей.
+ На всех этапах основные вычисления можно вынести в фоновый режим.
+ Все этапы легко масштабируемы как по пользователям, так и по товарам.
+ Все этапы должны быстро работать в онлайн-режиме.
+ Мгновенная реакция на проставление новых рейтингов.
+ Отсутствие необходимости наличия векторной модели профиля пользователя и описания товара на этапах 1–2.
+ Достаточно быстрая реакция на изменение профиля пользователя (рейтинги меняются сразу, а вот формирование начального списка требует вычислений близости с остальными пользователями, хотя это не очень затратно).

– Отсутствие данных об апробированности склейки векторов, производимой на этапе Estimation.

Литература

1. *Bennett J., Lanning S.* The Netflix Prize // KDD Cup Workshop at SIGKDD-07, 13th ACM Conference (International) on Knowledge Discovery and Data Mining Proceedings. — San Jose, California, USA, 2007. P. 3–6.
2. *Jannach D., Zanker M., Felfernig A., Friedrich G.* Recommender systems: An introduction. — Cambridge: Cambridge University Press, 2010.

3. *Ricci F., Rokach L., Shapira B., Kantor P. B.* Recommender systems handbook. — New York: Springer-Verlag, 2010.
4. *Schafer J. B., Frankowski D., Herlocker J., Sen S.* Collaborative filltering recommender systems // The Adaptive Web. — Berlin/Heidelberg: Springer, 2007. P. 291–324.
5. *Pazzani M. J., Billsus D.* Content-based recommendation systems // The Adaptive Web. — Springer Verlag, 2007. P. 325–341.
6. *Sahlgren M.* An introduction to random indexing // Methods and Applications of Semantic Indexing Workshop at the 7th Conference (International) on Terminology and Knowledge Engineering. — Citeseer: TKE, 2005.
7. *Linden G., Smith B., York J.* Amazon.com recommendations: Item-to-item collaborative filtering // Internet Computing, IEEE, 2003. Vol. 7. P. 76–80.