Рекомендательная система

©Ракитин Виталий Павлович,
МГУ им. М.В.Ломоносова, 5 курс, механико-математический факультет.
15 сентября 2016

Содержание

1 Цель

Проектировка рекомендательной системы.

2 Постановка задачи

Что мы имеем?

1. Множество пользователей $u = u(\Psi) \in U(\Psi) = \{u_1(\Psi), u_2(\Psi), \dots\}$, где Ψ — множество параметров, характеризующих данного пользователя:
   - Пол;
   - Возраст;
   - Социальный статус, социальная группа;
   - Регион, где он проживает;
   - Увлечения, интересы;
   - ...
2. Множество объектов $obj \in \Omega = \{obj_1, obj_2, \dots\}$;
3. Множество действий над объектами $r_{ij} = (i = u, j = obj) \in \Lambda = \{r_{11}, r_{12}, \dots\}$
   (оценка, покупка, просмотр и тд — **рейтинг**).

Замечание 2.1. *Из множества действий можно так же генерировать элементы множества Ψ (пользователи оптимисты, пользователи со схожими препочтениями ...)*

Что мы хотим получить?

Предсказать предпочтения конкретного пользователя $u \in U$, а так же составить для него **персональные рекомендации** на основе этих предпочтений.

3 Сбор информации о пользователе

Чем больше информации мы будем иметь — тем больше вероятность получить точное предсказание.

- При регистраиции предложить заполнить анекту с минимальной информацией о пользователе;
- Предложить синхронизацию социальных сетей;
- Отслеживать параметры «друзей», а так же их предпочтения;
- Следить за «явной» активностью пользователя (покупки, лайки);
- Фиксировать «неявную» активность (какие страницы посещает, на каких задерживается).

4 Введение связей между объектами

Идея: Нет смысла предлагать пользователям товар, который он уже преобрёл (оценил) или строго аналогичный, однако можно рекомендовать «родственные» объекты, например

- *Если* **u** *только преобрёл iPhone 6s, то нет смысла предлагать ему телефоны, но можно предложить аксессуары*;
- *Бессмысленно предлагать читать пользователю одну и ту же запись, опубликованную в разных местах, но можно предложить записи схожей тематики.*
- ...

Для каждого объекта введём 2 параметра:

1. **Тип;**
2. **Родство**.

Одинаковые (аналогичные) объекты нумеруются одинаковым **типом**, «близкие» — **родством**. Родство может задаваться как **«базовое»** (аксессуары, товары одного производителя/автора, с помощью хэш-тегов и тд), а так же его можно определять из соображения, какие объекты обычно оцениваются (преобретаются) вместе. После этого проведём кластеризацию всех объектов по степени родства.

5 Способы хранения данных

5.1 Объекты

Объект(Ω)	origID	Keys	TopObj	Тип	Родство			Time	Click
					subID	similarity	TopObj		

- **origID** — кодовый номер объекта;
- **Keys** — ключевые слова, хэш-теги, параметры, по которым определяется степень родства с другими объектами;
- **Click** — количество обращений;
- **Time** — время последнего обращения к объекту;
- **TopObj** — популярность данного объекта:

$$TopObj = function(Time, Click)$$

- **Родство** — связь с другими объектами;
- **Similarity** — степень родства объектов;
- **subID** — кодовый номер родственного объекта
(отсортированы по степени родства, в случае одинаковой — по Top);

5.2 Top-популярных объектов

Отдельно храним список объектов, отсортированный по популярности, на случай абсолютно холодного старта.

origID	Click

- **origID** — кодовый номер объекта;
- **Click** — количество обращений;

5.3 Пользователи

User (U)	uID	origID_i	r_ui	... Параметры(Ψ) ...	ObjTopID

- **uID** — уникальный номер пользователя;
- origID_i — все объекты, с которыми пользователь взаимодействовал;
- r_ui — рейтинг данных объектов;
- **Параметры** — все параметры которые мы смогли узнать о нашем пользователе;
- **ObjTopID** — Top-10 наиболее релевантных рекомендаций для данного пользователя;

5.4 База сходства пользователей

	u_1	u_2	u_3	u_4	...
u_1					
...					

- По строкам и столбцам распределены все пользователи конкретного кластера;
- На пересечении — степени их похожести $sim(u_i, u_j) = \frac{1}{1+dH(u_i,u_j)}$ (через растояние Хэмминга).

5.5 База предпочтений

Для каждого отдельно взятого пользовательского кластера строим таблицу следующего вида:

	obj_1	obj_2	obj_3	obj_4	...
$middle$	4		5	6	
u_1	3		7	9	
u_2	6	1	8	3	
u_3			2		
...					

- По строкам распределены все пользователи данного кластера;
- По столбцам — список всех объектов, с которыми взаимодействуют данные пользователи, а так же по N первых родственных объектов другого типа по степени родства.
- На пересечении расположены **рейтинги**, полученные от соответствующих пользователей соответствующими объектами(Λ). На местах пропуска впишем **оценку рейтинга** (наше «предсказание»).
- В строке **middle** указаны средние оценки объектов по всем пользователям кластера.

5.6 База кластеров пользователей

Номер кластера	uIDs	Центроид

- uIDs — списки пользователей в данном кластере;

6 Роботы

Определение 6.1. *Роботы — программы-демоны, обеспечивающие работу системы.*

1. **Кластеризатор пользователей** — проходит по базе (5.3), случайным образом выбирая начальные центры кластеров. Сохраняем J и Θ (ошибка и центроиды) для минимального J. Так как алгорит **k-means** является локальным, то проводим данную операцию большое количество раз;
2. **Редактор базы кластеров** — запускается, когда **кластеризатор Users** нашёл разбиение с меньшей ошибкой;
3. **Редактор базы пользователей сходства** — запускается, когда **редактор базы кластеров** изменил список кластеров и их состав;
4. **Редактор базы объектов** — добавляет новые объекты в базу (5.1), пересчитывает Родство объетов, определяет тип;
5. **Информатор** — при выполнении пользователями действий над объектами увеличивает **Click** и **TopObj**, изменяет **Time**, добавляет в базу (5.3) информацию о взаимодействии объектов, повышает количество **Click** в (5.2);
6. **Сортировщик Top** — регулярно сортирует базу популярных объектов;
7. **Редактор базы рекомендаций** — запускаетя при изменении базы сходства пользователей или базы объектов, высчитывает Top-10 рекомендаций для каждого пользователя;
8. **Редактор базы пользователей** — получает от **редактора базы рекомендаций** Top-10 и заносит его в базу (5.3);
9. **Мусорщик** — следит за **Time** в списке объектов и удаляет устаревшие, а так же все упоминания о них в других базах.

7 Построение рекомендаций

7.1 Кластеризация пользователей

Идея: похожим пользователям обычно нравятся похожие объекты, поэтому на основе пункта (3) кластеризуем множество U.

- Метрика — расстояние Хэмминга, а именно $dH(x, y)$ — количество различных компонент в **x** и **y**;
- Применим метод кластеризации **k-means**;
- Количество кластеров будем определять исходя из количества пользоваталей;
- Сохраним центроиды наших кластеров в множество $\Theta = \{c_1, c_2, \dots\}$.
- Мера ошибки
$$\overline{J} = \sum_{n=1}^{N} \sum_{k=1}^{k} r_{nk} d(x_n, \mu_k),$$
где $d(x_n, \mu_k)$ — функция расстояния, μ_k — один из объектов кластера.

7.2 Предсказание

Рассмотрим 2 подхода построения предсказания. Оба запроса будут выполняться на разных серверах. В качестве результирующего будем брать тот, который приходит быстрее, либо среднее арифметическое результатов.

7.2.1 Пользовательский

$$\overline{r}_{ui} = r_u + \frac{\sum\limits_{u \in U_i} sim(u, v) \cdot (r_{vi} - r_v)}{\sum\limits_{u \in U_i} sim(u, v)}$$

- r_u — средняя оценка нашим пользователем всех объектом;

7.2.2 Объектный

$$\overline{r}_{ui} = r_i + \frac{\sum\limits_{j \in I_u} sim(i,j) \cdot (r_{ui} - r_j)}{\sum\limits_{j \in I_u} sim(i,j)}$$

- r_j — средняя оценка пользователями одного кластера данного объекта;

7.3 Конечный алгоритм подбора рекомендаций

Рассмотрим несколько вариантов восприятия пользователя в системе:

1. *Пользователь существует в базе, информация по нему собрана:*
   - *Существует список Top-10 рекомендаций* — рекомендуем первые объекты из топа;
   - *Списка рекомендаций нет* — возьмём выборку пользователей из того же кластера из базы предпочтений, на их основе сделаем предсказания. Отсортируем предсказания по рейтингу и предложим Top;
2. Пользователя нет в базе:
   - *Можно собрать «неявную» информацию* — определим ближайшую группу из базы кластеров, возьмём выборку пользователей из этого кластера из базы предпочтений, на их основе сделаем предсказания. Отсортируем предсказания по рейтингу и предложим Top;
   - *Холодный старт* — предложить Top-популярных;