Практическое задание «Рекомендательные системы»

В этой работе мы будем решать задачу рекомендации музыки. То есть мы поставим целью получить модель, которая для каждого пользователя будет возвращать набор треков, наиболее похожих на те, что он уже слушал. В первой части мы разберемся с memory-based подходом и моделью со скрытыми переменными. Это не очень мощные методы, но зато они позволяют строить предсказания почти моментально. Затем, во второй части, мы обратим внимание на то, что датасет содержит огромное число треков и воспользуемся результатами уже построенных быстрых моделей для сокращения списка кандидатов до разумного количества. После этого проведем ранжирование среди кандидатов с помощью сильной, но чуть более медленной модели, и отберем самые лучшие варианты.

Необходимые файлы для выполнения практической доступны по ссылке

Задание 1 (4 балла)

Реализуйте подсчёт метрики MAP@k (слайды 130-133). Для этого реализуйте функции _compute_binary_relevance (1 балл), ap_at_k (2 балла) и map_at_k (1 балл) в utils/metrics.py

$$MAP@k = \frac{1}{N}\sum_{u=1}^{N} AP_u@k$$

$$AP_u@k = \frac{1}{\min(k, n_u)}\sum_{i=1}^{k} r_u(i)p_u@i$$

$$p_u@k = \frac{1}{k}\sum_{j=1}^{k} r_u(j)$$

N - количество пользователей.
n_u - число релевантных треков пользователя u на тестовом промежутке.
$r_u(i)$ - бинарная величина: относится ли трек на позиции i к релевантным.

Коллаборативная фильтрация (User2User)

Идея: чтобы выбрать треки, которые понравятся пользователю, можно набрать несколько похожих на него пользователей (соседей) и посмотреть, какие

треки они слушают. После этого остается агрегировать треки этих пользователей и выбрать самые популярные. Соответственно, задача состоит из двух частей: выбора функции похожести двух пользователей и способа агрегации.

В качестве функции похожести мы будем использовать меру Жаккара:

$$s(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$$

Во всех формулах:

- I_u - множество треков, прослушанных пользователем u.
- r_{ui} - прослушал ли пользователь u трек i (0 или 1).

Множество соседей определим как

$$N(u) = \{v \in U \setminus \{u\} \mid s(u, v) > \alpha\},$$

где $\alpha -$ гиперпараметр.

Для агрегации мы будем пользоваться следующей формулой.

$$\hat{r}_{ui} = \frac{\sum_{v \in N(u)} s(u, v) r_{vi}}{\sum_{v \in N(u)} |s(u, v)|}$$

Рекомендация треков в такой модели для конкретного юзера будет выглядеть как сортированный по убыванию $\hat{r}_{ui}$ список индексов треков.

Задание 2 (1 балл)

Реализуйте функцию подсчета меры Жаккара (jaccard_sim из utils/distances.py). Функция принимает матрицу оценок и вектор оценок пользователя u и возвращает вектор со значениями похожести пользователя u на всех пользователей.

Задание 3 (4 балла)

Реализуйте методы similarity (1 балл) и get_items_scores (3 балла) (вычисляет $\hat{r}_{ui}$ из описания выше) класса User2User из utils/models.py. recommend возвращает индексы треков, отсортированные в порядке убывания предсказанных оценок.

Задание 4 (4 балла)

Реализуйте функции _als_user_step (2 балла) и _als_item_step (2 балла) из utils/models.py

Задание 5 (8 баллов)

Реализуйте метод fit в классе ALS из utils/models.py. При реализации метода вы можете дописывать в класс ALS необходимые вам методы, общий критерий - лишь бы тесты корректно отбегали.