СОРЕВНОВАНИЯ

ЗАДАЧИ ОТОСЛАТЬ МОИ ПОСЫЛКИ СТАТУС ПОЛОЖЕНИЕ ЗАПУСК

А. Рекомендательная система

Есть некоторый новостной сайт, на котором представлены разные статьи. Сейчас здесь нет какой-то серьезной рекомендательной системы, пользователю показываются все недавние статьи по популярности.

В вашем распоряжении имеются данные о том, какие статьи ($item_id$) открывал пользователь ($user_id$) и порядок (order), в котором они были открыты. Порядок начинается с 1 (самое раннее действие) и увеличивается со временем.

Обучающая выборка представлена тремя колонками [user_id, item_id, order], где user_id показывает идентификатор пользователя, item_id идентификатор страницы и order — порядковый номер посещенной страницы.

user id, item id, order

0,0,1

0,1,2

0,2,3

Кроме того, в вашем распоряжении имеются описания пользователя. Колонки 0-15 — обезличенная информация о пользователе, колонка $user_id$ — ключ, по которому можно объединить данные по просмотрам и фичи пользователя. На основе предоставленных данных, вам необходимо предсказать следующие 3 сайта, которые посетит каждый пользователь из тестового выборки. Аналогично тренировочным данным, порядок в ней начинается с 1 и увеличивается со временем: order=1 означает, что это следующий посещенный сайт после окончания последовательности в обучающей выборке.

user_id,order

0,1

0,2

0,3

Файл sample-submission.csv содержит пример решения для загрузки в тестовую систему. В вашем сабмите должны быть колонки с пользователем $user_id$, предсказаниями айтема для этого пользователя $item_id$ и колонка order, описывающая порядок айтемов, в котором пользователь их увидит. Не забывайте, что порядок предсказаний в сабмите должен начинаться с 1 и увеличиваться с каждым следующим предсказанием.

user_id,item_id,order

0,1,1

0,1,2

0,1,3

В качестве метрики для оценивания качества используется метрика ранжирования mean average precision at k (MAP@k) при k=3. То есть в метрике будут участвовать только 3 первых предсказания. На лидерборде показывается метрика умноженная на 10000, то есть результат будет выглядеть как 10000 * MAP@k.

Расчет данной метрики можно представить в три этапа:

1. Считаем precision at k:

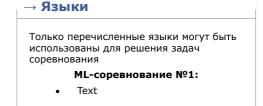
$$Precision@k = rac{1}{k} \sum_{i=0}^{k} (y^i_{true} == y^i_{pred})$$

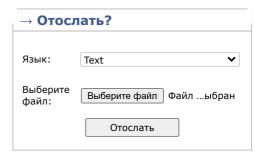
2. Считаем average precision at k:

$$AP@k = rac{1}{k}\sum_{i=0}^{k}(y_{true}^{i} == y_{pred}^{i})*Precision@i$$











3. Берем среднее от average precision at k:

$$MAP@k = rac{1}{N}\sum_{i=0}^{N}AP@k$$

Чтобы реализовать данную функцию локально можно использовать следующий алгоритм.

Ha вход нам приходят два pd.DataFrame gt и preds с колонками [user_id, item_id, order]. gt – ground truth, то есть настоящие данные, а preds – наши предсказания. Тогда мы можем использовать следующие шаги для расчета метрики.

- 1. Отфильтровать gt и preds по колонке order, чтобы его значение было не больше k (порядка k в MAP@k).
- 2. Сджойнить два датафрейма по user_id и order. Здесь нам важно сохранить информацию о том, какой item id на каком месте был для каждого пользователя.

После этих шагов мы готовы к расчету метрик по трем вышеописанным шагам для расчета $\mathtt{MAP@k}$

- 3. Добавим в объединенный датафрейм (joined) колонку is_right, показывающую, что на текущем месте ответ был правильным. 4. Так как для расчета precision@k нам нужно знать количество всех правильных событий до текущего момента, то посчитаем кумулятивную сумму колонки is_right в разрезе по пользователям. Назовем ее is right cum.
- 5. Теперь посчитаем precision@k поделив колонку is_right_cum на order. То есть разделим количество правильно угаданных событий на количество всех событий до текущего момента.
- 6. В AP@k участвуют только те precision@k, для которых в текущий момент времени был правильно угадан ответ. Поэтому мы можем добавить еще одну колонку p@k masked, которая является перемножением p@k на is right.
- 7. Посчитаем среднее в разрезе каждого пользователя по колонке p@k_masked, получим AP@k для каждого пользователя.
- 8. Теперь возьмем глобальное среднее по AP@k и получим итоговую метрику MAP@k.

Ниже представлен пример имплементации данного алгоритма, который можно использовать для локальной валидации.

```
def calc_map_k(gt: pd.DataFrame, preds: pd.DataFrame, k=3) -> float:
    # filter first k elements of ground truth
    gt = gt[gt.order <= k]</pre>
   # filter first k elements of predictions
   preds = preds[preds.order <= k]</pre>
    # join ground truth and predictions by user_id and order, fill missed
values from predictions by some non-existent value
    joined = gt.merge(preds, how="left", on=['user_id',
"order"]).fillna(-12345)
   # create indicator of right predictions
    joined["is_right"] = (joined.item_id_x == joined.item_id_y).astype(int)
    # calculate cumulative sum of all right predictions before current order
    joined["is_right_cum"] = joined.groupby("user_id").is_right.cumsum()
    # normalize it by order (precision@k)
    joined["p@k"] = joined["is_right_cum"] / joined["order"]
    # add relevance mask
    joined["p@k_masked"] = joined["p@k"] * joined["is_right"]
    # calculate mean user based (average precision @ k)
    ap = joined.groupby("user_id")["p@k_masked"].mean()
    # calculate mean average precision @ k
    return ap.mean()
```

Соревнования по программированию 2.0 Время на сервере: $13.08.2021\ 10:33:35\ (j1)$. Десктопная версия, переключиться на мобильную. Privacy Policy

На платформе

