687.1
Рейтинг

OTUS

Цифровые навыки от ведущих экспертов

MaxRokatansky 2 мая в 15:58

Метрики оценки для рекомендательных систем

5 мин 3.6K

Блог компании OTUS, Машинное обучение*

Перевод

Автор оригинала: Claire Longo

Рекомендательные системы становятся все более популярными в онлайн-ритейле благодаря своей способности предлагать персонализированный опыт определенным пользователям. Mean Average Precision at K [именованная средняя точность на K-элементах] (MAP@K) обычно является наиболее предпочтительной метрикой для оценки производительности рекомендательных систем. Однако использование дополнительных диагностических показателей и визуализаций может дать более глубокое и иногда весьма необычное представление о работе модели. В данной статье рассматриваются метрики *Mean Average Recall at K (MAR@K), Coverage, Personalization u Intra-list Similarity,* и на их основе проводится сравнение трех простых рекомендательных систем.

Если вы хотите использовать любую из метрик или графиков, рассмотренных в этой статье, я сделала их доступными в библиотеке python recmetrics.

```
$ pip install recmetrics
```

Датасет Movielens

Данные, используемые в этом примере, представляют собой популярный датасет Movielens 20m. Они содержат рейтинги фильмов, выставленные пользователями, а также теги жанров. (Для сокращения времени обучения объем этих данных был уменьшен, и они включали только оценки пользователей, которые проанализировали более 1000 фильмов, а результаты с рейтингом менее 3 звезд были удалены).

userId	movieId	rating
156	1	5.0
156	2	5.0
156	4	3.0

Пример пользовательского рейтинга фильмов

Модели

Были протестированы и сопоставлены три различные рекомендательные системы.

1. Случайный рекомендатель (рекомендует 10 случайных фильмов каждому пользователю)
2. Популярный рекомендатель (рекомендует 10 самых популярных фильмов каждому пользователю)
3. Коллаборативный фильтр (метод матричной факторизации с использованием SVD)

Давайте углубимся в метрики и диагностические графики, а затем проведем сравнение этих моделей!

График "длинный хвост" (Long Tail Plot)

Я люблю начинать каждый рекомендательный проект с просмотра *графика "длинного хвоста"*. Он используется для изучения закономерностей популярности в данных о user-item интеракции (взаимодействие пользователя с элементом), таких как клики, рейтинги или покупки. Как правило, только небольшой процент элементов имеет большой объем взаимодействий, и это называется "головой". Большинство элементов находятся в "длинном хвосте", но их доля среди общего числа интеракций невелика.

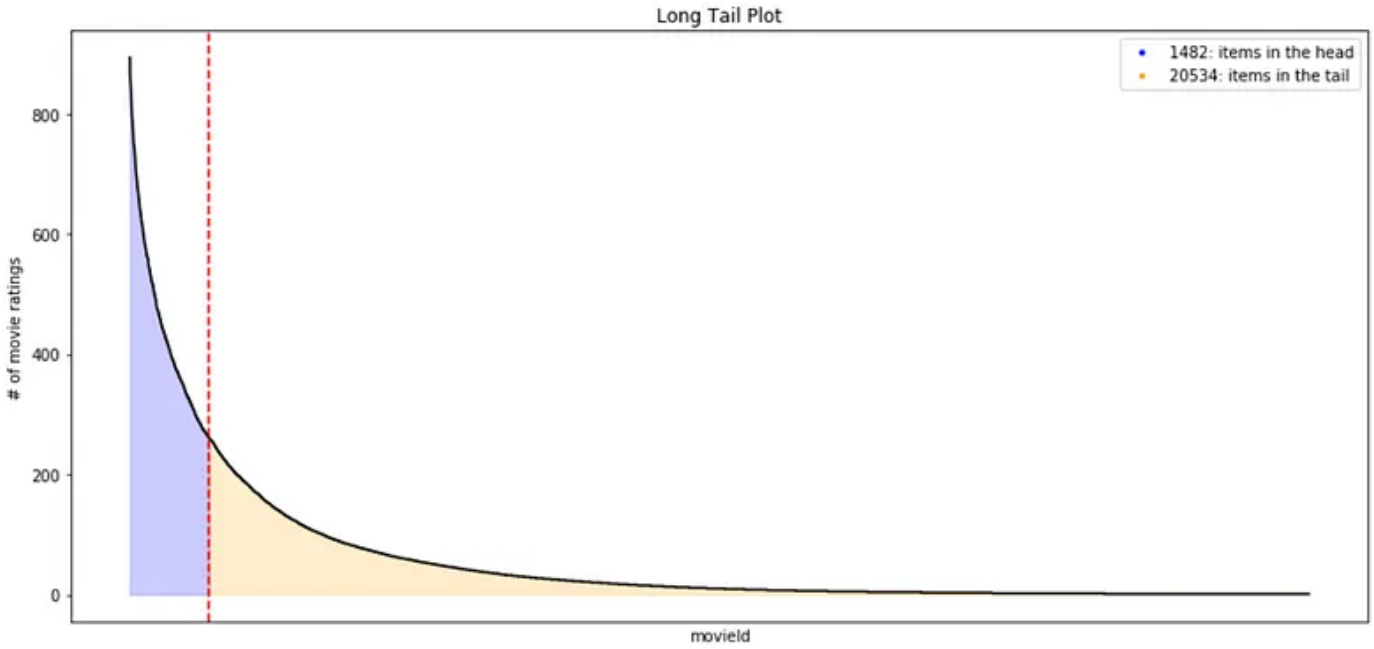


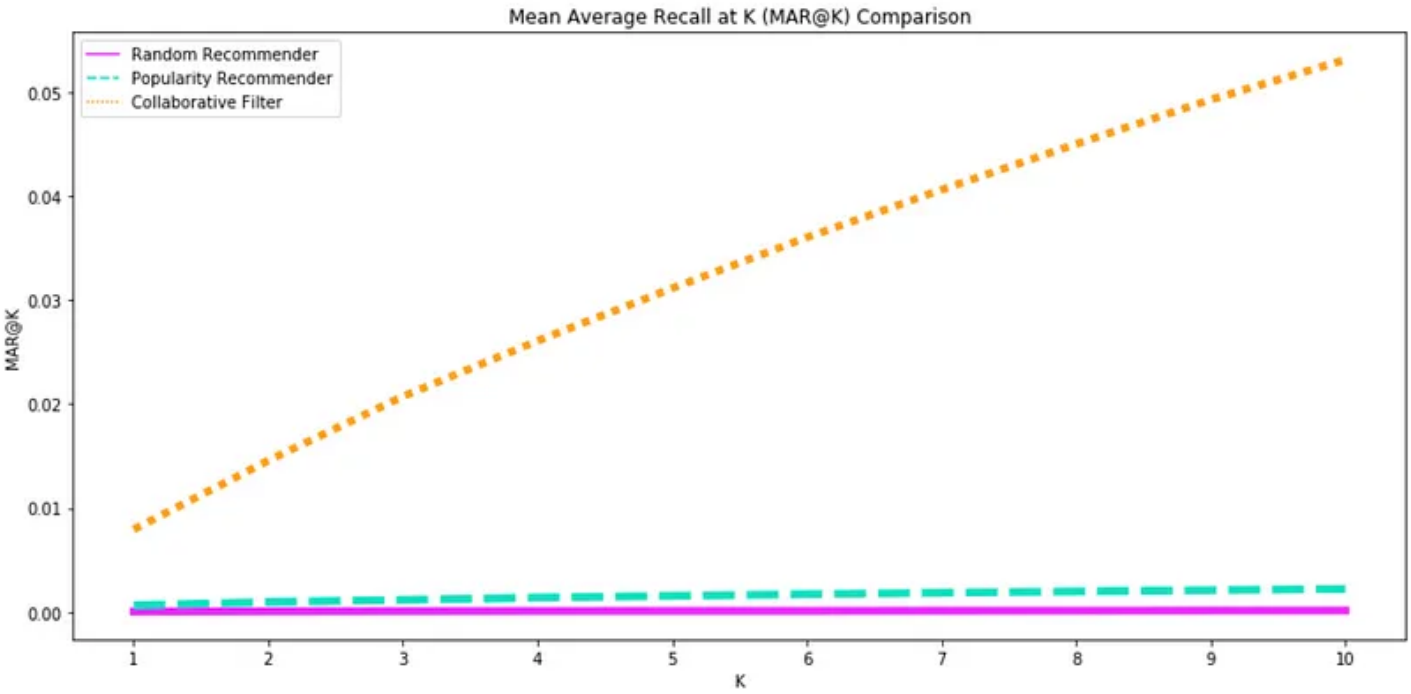
График “длинного хвоста”. (Пример данных о рейтингах в Movielens 20m)

Поскольку обучающие данные включают в себя множество результатов наблюдений за популярными элементами, для рекомендательной системы не составит труда научиться точно их предсказывать. В датасете фильмов самые популярные - это блокбастеры и классика. Большинству пользователей эти фильмы уже хорошо известны, и поэтому рекомендации по ним не смогут надлежащим образом удовлетворить их индивидуальные потребности или помочь обнаружить новые, подходящие варианты. Релевантные рекомендации определены в качестве рекомендаций элементов, которые пользователь положительно оценил в тестовых данных. Указанные здесь метрики обеспечивают методы оценки как релевантности, *так и* практической пользы рекомендаций.

MAP@K и MAR@K

Рекомендательная система обычно составляет упорядоченный список рекомендаций для каждого пользователя в тестовом наборе. MAP@K дает представление о том, насколько релевантен список рекомендуемых элементов, в то время как MAR@K — о том, насколько хорошо рекомендатель способен вспомнить все те элементы, которые пользователь положительно оценил в тестовом наборе. Я не буду подробно останавливаться на описании MAP@K и MAR@K, его можно найти здесь: Mean Average Precision (MAP) For Recommender Systems.

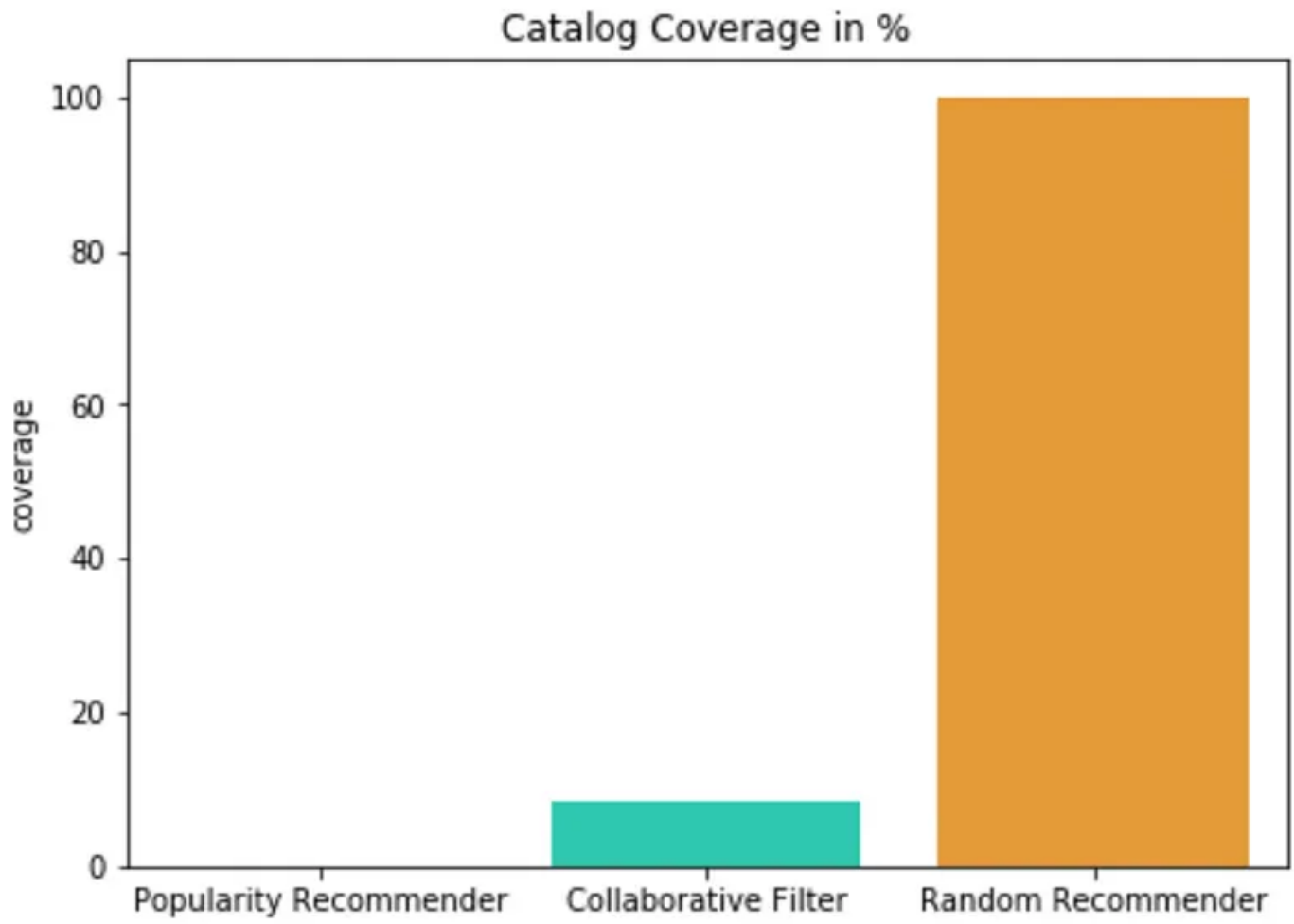
MAP@K доступен в библиотеке ml_metrics, а MAR@K я сделала доступным в recmetrics.



Благодаря MAR@K, коллаборативный фильтр способен вспомнить релевантные элементы для пользователя лучше, чем другие модели.

Покрытие (Coverage)

Покрытие — это процент элементов обучающих данных, которые модель может рекомендовать для набора тестов. В данном примере рекомендатель популярности имеет покрытие всего 0,05%, поскольку он всегда рекомендует только 10 элементов. Как и ожидалось, случайная рекомендация имеет почти 100% покрытие. Удивительно, но коллаборативный фильтр способен рекомендовать только 8,42% элементов, на которых он был обучен.



Сравнение покрытия для трех рекомендательных систем.

Персонализация (Personalization)

Персонализация — это отличный способ оценить, рекомендует ли модель одни и те же элементы разным пользователям. Это несходство (1- косинусное сходство) между списками рекомендаций пользователю. О том, как рассчитывается персонализация, лучше всего проиллюстрировано на примере.

```
example_predictions = [
    ['A', 'B', 'C', 'D'],
    ['A', 'B', 'C', 'X'],
    ['A', 'B', 'C', 'Z']
]
```

Пример списка рекомендуемых элементов для 3 разных пользователей.

	A	C	B	D	X	Z
0	1	1	1	1	0	0
1	1	1	1	0	1	0
2	1	1	1	0	0	1

Во-первых, рекомендуемые элементы для каждого пользователя представлены в виде бинарных индикаторных переменных (1: элемент был рекомендован пользователю. 0: элемент не был рекомендован пользователю).

```
[1.   , 0.75, 0.75]
[0.75, 1.  , 0.75]
[0.75, 0.75, 1.  ]
```

Затем рассчитывается матрица косинусного сходства через все векторы пользовательских рекомендаций.

$$personalization = 1-0.75 = 0.25$$

Наконец, вычисляется среднее значение верхнего треугольника косинусной матрицы. Персонализация равна 1 - среднее косинусное сходство.

Высокий показатель персонализации указывает на то, что рекомендации пользователям отличаются друг от друга. Это означает, что модель предлагает индивидуальный подход для каждого пользователя.

Сходство внутри списка (Intra-list Similarity)

Сходство внутри списка — это среднее косинусное сходство всех элементов в списке рекомендаций. Здесь для вычисления сходства используются характеристики рекомендуемых элементов (например, жанр фильма). Этот расчет также лучше всего проиллюстрировать на примере.

```python
example_predictions = [
    [3, 7, 5, 9],
    [9, 6, 12, 623],
    [7, 894, 6, 623]
]
```

Пример рекомендаций идентификаторов фильмов для 3 разных пользователей.

movieId	Action	Comedy	Romance
3	0	1	0
7	0	1	0
5	0	1	0
9	1	0	0

Эти характеристики жанра фильма используются для расчета косинусного сходства между всеми элементами, рекомендованными пользователю. В данной матрице показаны характеристики по всем рекомендованным фильмам пользователю 1.

Сходство внутри списка может быть рассчитано для каждого пользователя и усреднено для всех пользователей в тестовом наборе, чтобы получить оценку внутрисписочного сходства для модели.

```
recmetrics.intra_list_similarity(example_predictions, feature_df)
```

```
0.27777777777777773
```

Если рекомендательная система рекомендует отдельным пользователям списки очень похожих элементов (например, предлагаются только романтические фильмы), то сходство внутри списка будет высоким.

Использование правильных обучающих данных

Есть несколько вещей, которые можно сделать с обучающими данными, чтобы быстро улучшить рекомендательную систему.

1. Удалить популярные элементы из обучающих данных. (Это подойдет в тех случаях, когда пользователи самостоятельно смогут их найти и, как следствие, не сочтут такие рекомендации полезными).

2. Масштабировать рейтинги элементов в зависимости от ценности пользователя, например, по средней стоимости транзакции. Это может помочь модели научиться рекомендовать товары, которые приводят к появлению лояльных или наиболее ценных клиентов.

Заключение

Хорошая рекомендательная система дает релевантные и, приносящие практическую пользу, полезные рекомендации. Используя комбинацию нескольких оценочных показателей, можно начать анализ эффективности модели не только по релевантности. Посмотрите мою библиотеку python, если вы хотите использовать эти метрики и графики для оценки своих собственных рекомендательных систем.

Приглашаем всех желающих на открытое занятие «Рекомендательная система: как рекомендовать визуально похожие товары». На этом уроке вы узнаете, как сделать векторное представление изображений, поработаем с нейросетями компьютерного зрения, поищем похожие по фото объекты — украшения, предметы одежды. Обсудим, как это технически организовать для целей рекомендательной системы.

В результате вы узнаете, как работать с глубокими нейросетями компьютерного зрения, как готовить изображения для обработки, как сделать рекомендательную систему на основе

похожих изображений.

- Регистрация на открытый урок

Теги: рекомендательные системы, Метрики оценки, ds

Хабы: Блог компании OTUS, Машинное обучение

OTUS
Цифровые навыки от ведущих экспертов

Сайт ВКонтакте Telegram

82 Карма 102.8 Рейтинг

OTUS @MaxRokatansky
Редактор

Комментарии

Здесь пока нет ни одного комментария, вы можете стать первым!

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

AKlimenkov 21 час назад

Все дороги ведут к простым числам. Таинственная и удивительная история поиска самых совершенных чисел в мире