Отчет по первому заданию. Машинное обучение на больших данных.

Виктория Стренадко

17 ноября 2021 г.

1 Основная идея эксперимента

В данной ДЗ основной целью была реализация собственного рекоммендера и проведение A/B тестирования. Здесь мы сначала при помощи симулятора и системы, которая предлагает рандомные треки собрали и проанализировали пользовательские сессии, выбрав в качестве рейтинга время прослушивания пользователем трека. После обучения модели мы собрали рекомендации для каждого пользователя, которые использовали в новом алгоритме, и запустили эксперимент, суть которого состоял в том, что на разных группах мы использовали разные рекоммендеры и после окончания работы программы посмотрели на улучшение метрик по сравнению с контрольной группой.

2 Сбор данных для обучения модели и сбора рекомендаций

Так как на начальном этапе у меня возникли проблемы с утилитой `dataclient.py`, я взяла данные, которые выложил однокурсник на кластер в папку `/user/mob2021032/joined_dataset` (скопировав себе в папку `/user/mob2021083/data`), и обучала модель на них.

3 Алгоритм для подбора рекомендаций

В качестве алгоритма для построения рекомендаций я выбрала метод коллаборативной фильтрации `ALS(Alternating Least Squares)`, реализованный в питоновской библиотеке `pyspark.ml.recommendation`. После обучения моделей я сгенерировала список из 100 треков для каждого пользователя. Также я попробовала обучаться только на тех треках, которые пользователь прослушал более чем наполовину, но результат в дальнейшем оказался неудовлетворительным.

4 Модифицированный рекоммендер

В рамках работы над ДЗ я написала новый рекоммендер (Модуль `NewRecommender`). Главная идея заключается в том, что мы рекомендуем пользователю рандомный трек из выгруженного списка рекомендаций, но если получили такой же трек, какой был в прошлый раз, или для юзера не сгенерировалась рекомендация, то выбираем следующий, используя рекоммендер `StickyArtist`. Как видно из таблицы с анализом экспериментов, по среднему количеству прослушаных треков пользователем и среднему времени прослушивания, алгоритм `NewRecommender`(выборка `T1`) показывает наибольшую эффективность по сравнению с алгоритмами, реализованными на семинаре.

5 Разбиение на группы

Я разбила пользователей на 4 группы. Контрольная выборка - группа пользователей, которым рекомендуется рандомный трек. Для группы `T1` рекомендуется трек при помощи алгоритма `NewRecommender`, описанного выше. Для группы `T2` используется рекоммендер `StickyArtist`, для `T3` - `Contextual`.

6 Структура эксперимента и инструкция

Последовательность действий:

1. Склонировать репозиторий `https://gitlab.com/fpmi-atp/atp-mobod2021/-/tree/recsys/recsys` на локальную машину.
2. Зайти в папку `botify` в репозитории.
3. Собрать Docker-контейнер командой `docker-compose up -d -build`
4. Создать пустое окружение, активировать его, установить все зависимости для проекта, прописать пути.
5. Перейти в папку `sim` и запустить симулятор на день командой `python sim/run.py -episodes 1000 -recommender remote -config config/env.yml -seed 31337` . (Сбор данных описан в секции 1).
6. Собрать рекомендации для пользователей на кластере, запустив ноутбук `jupyter/DataAndModel.ipynb` (описание алгоритма можно посмотреть в секции 2).
7. Загрузить данные `users_recommendations.json` в папку `botify/data` на локальной машине.
8. Исправить код эксперимента в `botify` и добавить собственный рекоммендер.
9. Остановить командой `docker-compose stop` и собрать заново Docker-контейнер командой `docker-compose up -d -build`, запустить симулятор на день.
10. Так как я так и не смогла побороть утилиту, то вручную копировала логи контейнера командой `docker cp recommender-container:/app/log/. ./logs`, перекладывала их через протокол `sftp` на кластер, потом заходила по `ssh` на кластер, и перекладывала данные вручную в hdfs кластера. В репозитории данные лежат в папке `botify/logs/data_1711.json`
11. В ноутбуке `jupyter/Analyse_final.ipynb` проведен анализ A/B эксперимента и построены графики.

7 Результаты

Предложенный рекоммендер оказался самым эффективным с точки зрения удержания пользователей среди всех алгоритмов, предложенных на семинаре, что следует из сравнения метрик $mean\_session\_time$, $mean\_session\_length$.

	treatment	metric	effect	upper	lower	control_mean	treatment_mean	significant
6	T1	mean_session_time	160.164	193.59	126.739	2.12252	5.52204	True
10	T2	mean_session_time	125.315	139.951	110.678	2.12252	4.78235	True
11	T2	mean_request_time	58.0342	88.6343	27.4341	0.000750668	0.00118631	True
5	T1	mean_session_length	47.6297	57.8891	37.3703	7.13866	10.5388	True
9	T2	mean_session_length	37.5267	42.7257	32.3277	7.13866	9.81756	True
7	T1	mean_request_time	23.248	46.3767	0.119345	0.000750668	0.000925184	True
3	T3	mean_request_time	14.0575	34.8908	-6.77575	0.000750668	0.000856194	False
0	T3	sessions	-0.0744048	4.27694	-4.42575	1.05882	1.05804	False
4	T1	sessions	-0.15873	4.09496	-4.41242	1.05882	1.05714	False
8	T2	sessions	-2.05761	2.03381	-6.14904	1.05882	1.03704	False
1	T3	mean_session_length	-3.13052	1.16512	-7.42616	7.13866	6.91518	False
2	T3	mean_session_time	-9.92223	2.42829	-22.2727	2.12252	1.91192	False