ВМК

Задание 5. MapReduce. Коллаборативная фильтрация.

Практикум 317 группы, весна 2021-2022

Начало выполнения задания: 28 апреля 2022 года, 02:00.
Жёсткий Дедлайн: **23 мая 2022 года, 23:59.**

1 Введение. Рекомендательные системы

Сегодня рекомендательные системы встречаются повсеместно. В интернет-магазине Вы можете увидеть блоки с «похожими товарами», на новостном сайте «похожие новости» или «новости, которые могут вас заинтересовать», на сайте с арендой фильмов это могут быть блоки с «похожими фильмами» или «рекомендуем вам посмотреть».

Задача рекомендательной системы заключается в нахождении небольшого числа фильмов (`Item`), которые скорее всего заинтересуют конкретного пользователя (`User`), используя информацию о предыдущей его активности и характеристиках фильмов.

Широко известен конкурс компании `Netflix`, которая в 2006 году предложила предсказать оценки пользователя для фильмов в шкале от 1 до 5 по известной части оценок. Победителем признавалась команда, которая улучшит `RMSE` на тестовой выборке на 10% по сравнению с их внутренним решением. За время проведения конкурса появилось много новых методов решения поставленной задачи.

Обычно в таких задачах выборка представляет собой тройки $(u, i, r_{u,i})$, где u – пользователь, i – фильм, $r_{u,i}$ – рейтинг. Далее будем считать, что рейтинги нормализованы на отрезке $[0, 1]$.

1.1 Neighborhood подход в коллаборативной фильтрации

Имея матрицу user-item из оценок пользователей можно определить меру `adjusted cosine similarity` похожести товаров i и j как векторов в пространстве пользователей:

$$sim(i,j) = \frac{\sum\limits_{u \in U_{i,j}} (r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum\limits_{u \in U_{i,j}} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum\limits_{u \in U_{i,j}} (r_{u,j} - \overline{r_u})^2}} \tag{1}$$

где $U_{i,j}$ – множество пользователей, которые оценили фильмы i и j, $\overline{r_u}$ – средний рейтинг пользователя u.

Рейтинги для неизвестных фильмов считаются по следующей формуле:

$$\hat{r}_{u,i} = \frac{\sum\limits_{j \in I_u} sim(i,j) r_{u,j}}{\sum\limits_{j \in I_u} sim(i,j)} \tag{2}$$

где I_u - множество фильмов, которые оценил пользователь u. Такой подход называется `item-oriented`. Обратим внимание на то, что $sim(i,j) \in [-1, 1]$. Это может привести к делению на ноль или значениям $\hat{r}_{u,i}$ вне диапазона $[0, 1]$. Избавимся от этой проблемы, положив равными нулю отрицательные значения $sim(i,j)$.

2 Описание задания

В рамках данного задания Вам будет необходимо реализовать коллаборативную фильтрацию по формулам 1, 2 с использованием фреймворка `MapReduce`. Ваша программа, получая на вход список троек $(u, i, r_{u,i})$ и список соответствий между номером фильма и его названием, должна вывести для каждого пользователя **топ-100** фильмов с самым высоким предсказанным рейтингом.

При вычислений рекомендаций необходимо учитывать только те фильмы, которые пользователь ещё не оценил. Рекомендации выводятся по убыванию предсказанной оценки. При равенстве предсказанных оценок выше в списке рекомендаций должен стоять фильм с лексикографически меньшим названием.

Строки в файле с предсказаниями необходимо представить в следующем виде:

<user_id>@<rating_1>%<film_name_1>@...@<rating_100>%<film_name_100>

Рис. 1: Формат выходных данных

В качестве датасета предлагается использовать MovieLens. Используйте `small` версию датасета. Результат работы Вашего решения на этом датасете нужно приложить при сдаче задания (папка `data/output/final`).

При выполнении задания необходимо привести подробное описание Вашего решения в файле `description.md/html/pdf`. В частности:

1. Описание каждой стадии выполнения программы и каждой map-reduce задачи
2. Сложность по числу операций и по количеству памяти для каждого маппера и редьюсера. Используйте следующие обозначения: U – общее число пользователей, I – общее число фильмов, M – число мапперов, R – число редьюсеров, α – средняя доля фильмов, оценённых одним пользователем (эквивалентно средней доле пользователей, оценивших один фильм и доле известных оценок к общему числу возможных оценок UI)
3. Суммарное время работы вашей программы
4. Решение бонусных заданий

3 Технические детали реализации

Обратите внимание на следующие моменты, которые помогут успешно решить задачу:

- Пример запуска Hadoop Streaming программы на кластере
- Для использования пользовательских сепараторов используйте следующие опции:

```
-D stream.num.map.output.key.fields=<number_of_fields_for_key>
-D stream.map.output.field.separator=<custom_separator>
-D stream.reduce.input.field.separator=<custom_separator>
-D mapreduce.map.output.key.field.separator=<custom_separator>
```

- Для реализации вторичной сортировки могут пригодится следующие опции:

```
-D mapreduce.partition.keycomparator.options=<sort_options>
-D mapreduce.job.output.key.comparator.class= \
                org.apache.hadoop.mapred.lib.KeyFieldBasedComparator
-D mapreduce.partition.keypartitioner.options=<partition_options>
-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner
```

- Для управления памятью, выделяемой контейнерами используйте следующие опции:

```
-D mapreduce.map.memory.mb=<physical_memory_for_each_mapper>
-D mapreduce.reduce.memory.mb=<physical_memory_for_each_reducer>
-D mapreduce.map.java.opts=-Xmx<heap_size_for_each_mapper>m
-D mapreduce.reduce.java.opts=-Xmx<heap_size_for_each_reducer>m
-D yarn.nodemanager.vmem-pmem-ratio=<virtual_to_physical_memory_ratio>
```

  Подробнее смотрите в курсе на Stepik

- Детали этих опций и другие параметры смотрите в документации по Hadoop Streaming
- Учтите, что $sim(i, i) = 1 \; \forall i \in \{1, ..., I\}$. Также, гарантируйте отсутствие деления на ноль при вычислениях по формулам 1, 2, добавив к знаменателю небольшое число (например, 10^{-9})
- Используйте ровно `3` знака после запятой для передачи вещественных чисел между воркерами
- В файле `movies.csv` названия фильмов могут содержать запятые, поэтому используйте `csv.reader` из библиотеки `csv` для корректного разбиения строк этого файла.
- Задание можно реализовать так, чтобы время выполнение задачи было `≈ 30 минут`. Если Ваша программа выполняется сильно дольше, значит Ваша реализация неоптимальна. Неоптимальные реализации будут существенно штрафоваться

4 Формат сдачи задания

При выполнении задания необходимо использовать docker-контейнеры для запуска Hadoop кластера. Инструкцию по развертыванию контейнеров смотрите в репозитории. Другие форматы сдачи допускаются только по согласованию с преподавателем.

```
collaborative_filtering
├── data
│   ├── input ..................................Входные данные для MapReduce задачи
│   │   ├── rating.csv
│   │   └── movies.csv
│   └── output .....................................Результат всех MapReduce задач
│       ├── stage_1
│       │   ├── _SUCCESS
│       │   ├── part-00000
│       │   └── ...
│       ├── ...
│       └── final.............................................Итоговые рекомендации
│           ├── _SUCCESS
│           ├── part-00000
│           └── ...
├── src ..........................................Исходники MapReduce алгоритма
│   ├── mapper_1.py
│   ├── reducer_1.py
│   ├── ...
│   ├── mapper_n.py
│   └── reducer_n.py
├── description.md ..............................................Описание решения
├── run_hadoop.sh ..............Скрипт для запуска Hadoop Streaming на namenode
└── run.sh .....................................................Скрипт запуска MR задачи
```

Рис. 2: Требуемая структура решения

В результате, проект должен в точности удовлетворять структуре на диаграмме 2. В качестве решения необходимо сдать архив данной структуры с названием `<ФИО>_task_05.zip`.

Некоторые детали формата сдачи:

- Скрипт `run.sh`, обеспечивает полную работу задачи, начиная от загрузки данных внутрь docker-контейнера и запуска `run_hadoop.sh`, заканчивая копированием результата работы из docker-контейнера. Запуск программы должен подразумевать только запуск этого скрипта
- Файл `description.md` содержит описание Вашего решения. Все формулы в описании должны быть оформлены в виде LaTeX-уравнений. Допускается использовать `Markdown` синтаксис для оформления решения. Также, можете использовать `Jupyter Notebook`, сохранённый в формате `HTML`
- Папка `src` содержит код, используемый в задании. В частности, для каждой map-reduce стадии Вашей программы должны быть файлы `mapper_<stage_n>.py`, `reducer_<stage_n>.py`. Если используются другие скрипты (комбайнер и так далее) они также должны иметь в названии номер соответствующей стадии
- Папка `data/input`, содержит входные данные (файлы `ratings.csv`, `movies.csv`). При проверке решения входные файлы будут располагаться в этой директории
- В папке `data/output`, после завершения работы `run.sh` должны содержаться результаты работы программы. В частности, внутри директории `data/output.stage_<stage_n>` дожен находиться выход редьюсера (или маппера в случае map-only задачи) соответствующей стадии. В папке `final` должен быть итоговый список рекомендаций. Также, Вы можете предусмотреть сохранение любых промежуточных результатов в папке `data/output`. Присылать сами промежуточные файлы не требуется, необходимо прислать только содержимое папки `data/output/final`
- Разрешена обработка данных вне HDFS для дальнейшего их использования в Replicated Join
- Использование map-reduce стадий только с одним редьюсером запрещено

- Решение будет оценено в полный балл только при условии, что запуск `run.sh` на разных входных данных будет успешен

5 Бонусные задания

5.1 Анализ полученного решения (4 балла)

Выполните следующие пункты:

1. Реализуйте скрипт для генерации данных, похожих на настоящие. Реализуйте генерацию для различных U, I, α
2. Замерьте время работы каждой стадии программы для различных значений U, I, α, M, R. Обязательно сделайте замеры времени для датасета большего объема, чем в датасете `small`
3. Постройте графики зависимости времени работы от разных параметров
4. Докажите, используя полученные данные, что асимптотика Вашего решения соответствует теоретической (например, можно отдельно нарисовать графики в `log-log` шкале). Если наблюдается несоответствие, то объясните почему

5.2 Использование фреймворка (2 балла)

Реализуйте Вашу программу с использованием фреймворка mrjob.

5.3 Запуск на больших данных (2 балла)

Запустите задачу на существенно больших данных. Обратите внимание, выполнение может занять порядка ~~6-7~~14-15 часов и до 70Gb памяти. Засеките время работы каждой стадии и сравните со временем работы на `small` датасете. Согласуются ли полученные результаты с теоретическими формулами для сложности алгоритма? Если нет, то почему?