Задание 5. MapReduce. Коллаборативная фильтрация. Описание решения, Борисов Алексей, 317 группа.

Мое решение состоит из 5 MapReduce задач. Отдельно опишу каждую из них.

1 Задача 1

Цель данного этапа получить индекс следующего вида: $u \to \bar{r}_u, (i_1, r_{u,i_1}), \ldots, (i_n, r_{u,i_n})$. То есть для каждого пользователя получить список из фильмов, которые этот пользователь оценил, соотвествующие оценки, а также среднюю оценку пользователя $\bar{r}_u$.

В маппер поступают строки файла 'ratings.csv' следующего вида: $(u, i, r_{u,i}, \text{timestamp})$. Полученный рейтинг сразу нормализуем (далее будем использовать то же обозначение $r_{u,i}$). Строится отображение $u \to (i, r_{u,i})$.

- Сложность по памяти получается $O(1)$, так как мы не сохраняем ввод, а после преобразования сразу пробрасываем дальше.
- Сложность по числу операций выполненных на одном маппере равна $O(\alpha UI/M)$, так как каждый маппер обрабатывает такое число входных строк.

Стоит отметить, что необходимо пропустить первую строку файла 'rating.csv', которая представляет собой названия полей.

После сортировки и группировки по ключам входные данные редьюсера имеют следующий вид: $u \to [(i_1, r_{u,i_1}), \ldots, (i_n, r_{u,i_n})]$. Выполняем подсчет средней оценки пользователя $\bar{r}_u$. После чего можем осуществить вывод вида $u \to \bar{r}_u, (i_1, r_{u,i_1}), \ldots, (i_n, r_{u,i_n})$.

- Сложность по памяти получается $O(\alpha I)$, так как в некоторый момент времени необходимо сохранять все фильмы пользователя и соответствующие оценки.
- Сложность по числу операций выполненных на одном редьюсере равна $O(\alpha UI/R)$.

2 Задача 2

Цель данного этапа получить оценки похожести между фильмами: $(i_1, i_2) \to \text{sim}(i_1, i_2)$.

В маппер поступают строки вида $u \to \bar{r}_u, (i_1, r_{u,i_1}), \ldots, (i_n, r_{u,i_n})$ из выхода предыдующего этапа. Считаем разности $(\bar{r}_u - r_{u,i})$ для всех фильмов, которые далее будут использоваться в вычислении $\mathrm{sim}(i,j)$. Вообще говоря, в данной задаче эти разности можно было посчитать на предыдущем этапе, и не сохранять средний рейтинг пользователя, но в реальности он может нам потом еще пригодиться. Далее из списка фильмов получаем все неупорядоченные пары (i_1, i_2) и соответствующие им $(r_{u,i_1} - \bar{r}_u, r_{u,i_2} - \bar{r}_u)$. На выход из маппера подаем отображения следующего вида: $(i_1, i_2) \to (r_{u,i_1} - \bar{r}_u, r_{u,i_2} - \bar{r}_u)$, где $i_1 \leq i_2$ (для определенности).

- Вообще говоря, при получении всех пар в маппере никакой дополнительной памяти использовать не нужно, так как можно использовать генератор, поэтому сложность по памяти получается $O(\alpha I)$ — размер одной входной строки.

- Сложность по числу операций для одной строки равна квадрату от ее размера (так как число пар $n(n-1)/2$), то есть $O(\alpha^2 I^2)$, тогда суммарное число операций на одном маппере — $O(\alpha^2 I^2 U/M)$.

После сортировки и группировки по ключам в редьюсере происходит агрегация полученных данных и подсчет оценок схожести $\mathrm{sim}(i_1, i_2)$ по известным формулам. На выход из редьюсера подаем только положительные оценки похожести, так как остальные не влияют на предсказанный рейтинг. Выходные данные редьюсера имеют вид: $(i_1, i_2) \to \mathrm{sim}(i_1, i_2)$.

- Сложность по памяти получается $O(1)$, так как внутри редьюсера используются лишь несколько переменных для аккумулирования значений для подсчета $\mathrm{sim}(i_1, i_2)$, остальные данные не запоминаются.

- Сложность по числу операций для каждого ключа в редьюсере получается $O(|U_{ij}|)$, столько операций необходимо для подсчета одной оценки похожести. Введем дополнительные обозначения, β — число известных оценок похожести $\mathrm{sim}(i,j)$ и β^+ — число положительных оценок похожести $\mathrm{sim}(i,j)$ (Честно говоря, не придумал как эти величины выразить через предоставленные в задаче). Тогда сложность по числу операций для одного редьюсера равна $O(\beta |U_{ij}|/R)$.

Особенно непонятно как выразить через заданные величины β^+, так как при одинаковых значениях остальных величин, оно кажется может получаться существенно разным. **(?)**

3 Задача 3

Целью данного этапа является получение отображений следующего вида: $(u, j) \to (i, r_{u,i}, \text{sim}(i, j))$. Такое представление данных будет использовано в следующем этапе для предсказания рейтингов неоцененных фильмов $\hat{r}_{u,j}$.

На вход маппера подаются данные двух разных форматов: файл 'ratings.csv', формат которого был описан выше, и выход этапа 2, который имеет вид $(i_1, i_2) \to \text{sim}(i_1, i_2)$.

Для каждой строки из файла 'ratings.csv' формируется отображение $i \to \text{u}(u, r_{u,i})$, а для строк из этапа 2 формируются отображения $i_1 \to (i_2, \text{sim}(i_1, i_2))$ и $i_2 \to (i_1, \text{sim}(i_1, i_2))$. Символ 'u' нужен, для того чтобы определить, «откуда» пришло отображение.

- Сложность по памяти получается $O(1)$, так как внутри маппера никакие данные не запоминаются.
- Сложность по числу операций для одного маппера получается $O((\alpha UI + \beta^+)/M)$.

После сортировки и группировки по ключам в редьюсере мы аккумулируем список пользователей, которые оценили фильм i, а также список других фильмов j, для которых у нас есть оценки $\text{sim}(i, j)$. После чего осуществляется декартово произведение этих списков, с помощью которого формируются выходные данные следующего формата: $(u, j) \to (i, r_{u,i}, \text{sim}(i, j))$.

- Сложность по памяти получается $O(\alpha UI + \beta^+)$, столько данных соответствует одному ключу, их все нужно запоминать для осуществления декартового произведения. Сами элементы произведения запоминать не нужно, так как их можно отправлять на выход редьюсера с помощью генератора.
- Число операций для обработки одного ключа равно $O(\alpha UI \cdot \beta^+)$ (для обработки декартова произведения). Всего ключей I, поэтому суммарное число операций на одном редьюсере равно $O(\alpha UI^2 \cdot \beta^+/R)$

На данном этапе мы могли получить пары (u, j), для которых фильм j уже был оценен пользователем u. Учет этого момента будет осуществлен в следующем этапе.

Возможно здесь можно было реализовать более эффективно с использованием Repartition Join, но как-то быстро идея в голову не пришла, а большого количества времени нет.

4 Задача 4

Целью данного этапа является предсказание рейтингов $\hat{r}_{u,j}$ для неоцененных фильмов. То есть построение отображения $(u, j) \to \hat{r}_{u,j}$.

В маппере на вход поступает файл 'ratings.csv' и выход предыдущего этапа. Выход предыдущего этапа без изменений передается на выход из маппера, а длякаждой строки из 'ratings.csv' на выход подается отображение $(u, i) - >$ 'r'. Это делается для того, чтобы в редьюсере мы смогли определить, что для данной пары пользователь + фильм уже известна оценка и предсказывать ее не нужно.

- Сложность по памяти получается $O(1)$ так как никакие входные данные не запоминаются.
- Всего на входы мапперов передается $O(\alpha UI^2)$ строк, поэтому сложность по числу операций для одного маппера получается $O(\alpha UI^2/M)$.

После сортировки и группировки по ключам в редьюсере аккумулируются значения $\mathrm{sim}(i, j)$ и $r_{u,j}$ необходимые для подсчета предсказания $\hat{r}_{u,j}$. Если при обработке значений с одним ключом встретилась строка $(u, j) \to$ 'r', то для данного ключа не считается предсказание.

- Сложность по памяти получается $O(1)$ так как никакие входные данные не запоминаются. Для подсчета предсказания достаточно нескольких переменных, в которых аккумулируется необходимая информация.
- Всего на входы редьюсеров передается $O(\alpha UI^2)$ строк, поэтому сложность по числу операций для одного редьюсера получается $O(\alpha UI^2/R)$.

5 Задача 5

Целью данного финального этапа является получение списков рекомендаций в формате, указанном в задании.

На вход мапперу подается выход предыдущего этапа, в котором сформировались все предсказанные рейтинги $\hat{r}_{u,i}$. С помощью информации из файла 'movies.csv' из $(u, i) \to \hat{r}_{u,i}$ получаем $u \to (\mathrm{i\_name}, \hat{r}_{u,i})$, где i_name — название соответствующего фильма. В таком виде данные передаем на выход из маппера. Опять возникли проблемы с выводом сложности с

использованием вышевведенных величин. Пусть γ — среднее число фильмов, которые пользователь не смотрел, и при этом по ним посчитан $\hat{r}_{u,i}$. То есть по сути это количество среднее число фильмов, для которых посчитаны предсказания для пользователя. (это также равно числу входных строк поступающих на мапперы)

- Сложность по памяти получается $O(1)$ так как никакие входные данные не запоминаются.
- Всего на входы мапперов передается $O(\gamma U)$ строк, поэтому сложность по числу операций для одного маппера получается $O(\gamma U/M)$.

Далее можно было воспользоваться опциями для настройки вторичной сортировки и в редьюсере только собрать данные в нужный формат. (упорядочение по рейтингу и лексикографическому порядку произошло бы на этапе вторичной сортировки). Но я предпочел осуществить сортировку вручную в редьюсере.

После сортировки и группировки по ключу в редьюсере собираются все пары $(\hat{r}_{u,i}, \text{i\_name})$ и производится сортировка в необходимом порядке. Так как нам нужны только 100 фильмов с самым высоким рейтингом, то можно осуществить не обычную сортировку, а выделить 100 необходимых пар (например с помощью numpy.partition) и далее осуществить только сортировку этой части данных.

- Сложность по памяти получается $O(\gamma)$, так как необходимо сохранять данные соответствующие одному ключу.
- Сложность по числу операций на одном редьюсере при осуществлении полной сортировки — $O(U\gamma \log \gamma/R)$. Кажется, что при использовании partition можно сделать $O(U\gamma/R)$ (но в моем коде использована обычная сортировка).

Наибольшие проблемы возникли с оценкой сложностей разных этапов, со многими из них как видно из описаний выше, справиться не удалось.

6 Время работы

Суммарное время работы программы при запуске на 3 мапперах и 3 редьюсерах (такое количество использовалось на всех этапах) составило **36 минут**. Отдельно по этапам:

- Задача 1 — 30 секунд;
- Задача 2 — 3 минуты 30 секунды;
- Задача 3 — 4 минуты 50 секунд;
- Задача 4 — 22 минуты 30 секунд;
- Задача 5 — 4 минуты 30 секунд;

Как мы видим, большую часть времени занимает задача, реализующая предсказания оценок $\hat{r}_{u,i}$.