BIGDATA TEAM

HW #05: Spark RDD

автор задания: BigData Team, коллективная работа.

редактор задания[1]:

- Алексей Казюлин
- Big Data Mentor @ BigData Team

[1] Хочешь стать ментором и оставить след в истории Big Data? Тогда хорошо учись, помогай другим и дай нам знать о своем желании. Смело пиши автору задания или менеджеру учебного курса.

1. Описание задания

В данном ДЗ нужно решить **2 задачи**. Для решения использовать только Apache Spark RDD API. Цель: разобраться в принципах работы с RDD

2. Задача #1 (Task ID: spark.bigram): народные биграммы

Найдите все пары двух последовательных слов (биграмм), где первое слово:

```
narodnaya
```

Для каждой пары подсчитайте количество вхождений в тексте статей Википедии. Выведите все пары с их частотой вхождений в лексикографическом порядке. Формат вывода:

```
word_pair <tab> count
```

2.1. Входные данные

Дамп Википедии
`en_articles_part`:

- Путь на кластере: полный[2] датасет - `/data/wiki/en_articles_part`
- Формат: текст
- В каждой строке находятся следующие поля, разделенные знаком табуляции:
  1. INT - id статьи,
  2. STRING - текст статьи,

Пример:

```
12    Anarchism         Anarchism is often defined as a political philosophy which holds the state to be undesirable, unnecessary, or harmful.
```

Стоп-слова (**требуется только во 2-ой задаче**)
`stop_words_en`:

- Путь на кластере: `/data/stop_words/stop_words_en-xpo6.txt`
- Формат: одно стоп-слово на строчку

[2] Да, здесь нет ошибки, работаем на части данных, чтобы побыстрее познакомиться со Spark RDD

Пример:

```
...
wherein
whereupon
wherever
...
```

2.2. Выходные данные

Пример вывода:

```
...
narodnaya_zoo 42
narodnaya_apple 100500
narodnaya_country 10
narodnaya_love 777
...
```

2.3. Требования к реализации

1. PySpark-скрипты для запуска решений следует называть `task_<Surname>_<Name>_bigram.py`:
2. запустить с помощью команды: `PYSPARK_DRIVER_PYTHON=python3.6 PYSPARK_PYTHON=python3.6 spark-submit "task_<Surname>_<Name>_bigram.py"`
3. id статьи не учитывать в вычислениях;
4. для однозначности вычислений, выделить слова из статьи (включая название статьи) с помощью регулярного выражения re.findall(r"\w+", text);
5. стоп-слова не фильтровать;
6. привести все слова к нижнему регистру;
7. слова в паре объединить символом нижнего подчеркивания “_”;
8. отсортировать слова в выводе (STDOUT) по алфавиту;
9. Вывод (STDOUT) сохранить в файл: task_<Surname>_<Name>_bigram.out.[3];

Условие быстродействия:

- решение должно отрабатывать в течение 3х минут на свободном кластере (3 ноды x 8 CPU x 16GB RAM).

[3] Для подготовки архива с решением и выводом результатов запуска можно воспользоваться командой “tee”

3. Задача #2 (Task ID: spark.collocation): коллокации

Необходимо найти топ коллокаций в Википедии по метрике NPMI. Коллокация - это комбинации слов, которые часто встречаются вместе. Например: «high school» или «Roman Empire». Чтобы определить, является ли пара слов коллокацией, воспользуйтесь метрикой NPMI - нормализованная точечная взаимная информация.

Чтобы рассчитать NPMI, введем несколько определений:

1. `P(a)` - вероятность увидеть слово “a” в датасете.
   `P(a) = num_of_occurrences_of_word_"a" / total_number_of_words`
   `total_number_of_words` - общее количество слов в тексте
2. `P(ab)` - вероятность увидеть пару слов “a” и “b”, идущих подряд.
   `P(ab) = num_of_occurrences_of_pair_"ab" / total_number_of_word_pairs`
   `total_number_of_word_pairs` - общее количество пар
3. PMI(a,b) = ln(P(ab) / [P(a) x P(b)])
4. `NPMI(a,b) = PMI(a,b) / -ln(P(ab))` - величина PMI нормализованная в диапазон [-1, 1];

Примеры и комментарии:

- значение NPMI, равное “-1”, будет означать, что пара слов никогда не встречается в датасете. Например, такие пары, как “green idea” или “sleeps furiously”, никогда не встречаются вместе, поэтому `P(ab) = 0`, следовательно `PMI(a,b) = -inf, NPMI = -1`;
- значение NPMI, равное “0”, будет означать, что слова в паре встречается абсолютно независимо друг от друга. Рассмотрим пример "the doors": “the” может встретится рядом с любым словом. Таким образом, `P(ab) = P(a) x P(b)` и `PMI(a,b) = ln(1) = 0, NPMI = 0`;
- значение NPMI. равное “1”, будет означать, что это идеальная коллокация. Предположим, что "Roman Empire" - это уникальная комбинация, и за каждым появлением “Roman” следует “Empire”, и, наоборот, каждому появлению “Empire” предшествует “Roman”. В этом случае, `P(ab) = P (a) = P(b)`, поэтому `PMI(a,b) = -ln(P(a)) = -ln(P(b))`, следовательно NPMI = 1.

3.1. Входные и выходные данные

Используйте входные данные (Статьи Википедии и стоп-слова) из предыдущего задания.

Формат вывода:

```
word_pair <tab> npmi
```

Пример вывода:

```
...
south_africa      0.619
roman_empire      0.603
...
```

3.2. Требования к реализации

1. PySpark-скрипты для запуска решений следует называть `task_<Surname>_<Name>_collocation.py`:
2. запустить с помощью команды: `PYSPARK_DRIVER_PYTHON=python3.6 PYSPARK_PYTHON=python3.6 spark-submit "task_<Surname>_<Name>_collocation.py"`
3. id статьи не учитывать в вычислениях;
4. для однозначности вычислений, выделить слова из статьи (включая название статьи) с помощью регулярного выражения re.findall(r"\w+", text);
5. привести все слова к нижнему регистру;
6. удалить стоп-слова;
7. слова в паре объединить символом нижнего подчеркивания “_”;
8. отфильтровать биграммы, которые встретились не реже 500 раз (т.е. проводим все необходимые join'ы и считаем NPMI только для них, НО оценку вероятности встретить биграмму считаем на полном датасете). Общее число биграмм вычисляется после фильтрации стоп слов;
9. отсортировать слова в выводе по значению NPMI;
10. вывести (STDOUT) TOP-39 отсортированных по NPMI самых популярных коллокаций и их значения NPMI (округляем до 3-го знака после запятой, см. round);
11. Вывод (STDOUT) сохранить в файл: task_<Surname>_<Name>_collocation.out.[4];

Условие быстродействия:

- решение должно отрабатывать в течение 3-х минут на свободном кластере (3 ноды x 8 CPU x 16GB RAM).

[4] Для подготовки архива с решением и выводом результатов запуска можно воспользоваться командой “tee”

4. Критерии оценивания

Веса задач:

- spark.bigram - 40%
- spark.collocation - 60%

Балл за задачу складывается из:

- **60%** - правильное решение задачи
- **20%** - поддерживаемость и читаемость кода
  - в общем случае см. Clean Code и Google Python Style Guide
  - оценка качества будет проводиться автоматическим вызовом `pylint`:
    - `pylint *.py -d invalid-name,missing-docstring --ignored-modules=pyspark.sql.functions`
    - качество кода должно оцениваться выше 8.0 / 10.0
    - проверяем код Python версии 3 с помощью pylint==2.5.3
- **20%** - эффективность решения (такие как потребляемые CPU-ресурсы, скорость выполнения (в предположении свободного кластера)).

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения после soft deadline и до hard deadline
- **5%** за каждую дополнительную посылку в тестирующую систему (всего можно делать до 3-х посылок без штрафа):

Пример работы системы штрафов:

День	Посылка	Штраф
День 1	Посылка 1	Без штрафа
День 1	Посылка 2	Без штрафа
День 1	Посылка 3	Без штрафа
День 1	Посылка 4	-5%
День 2	Посылка 5	-5%
День 3	Посылка 6	-5%

Итоговый штраф: -15%

Для подсчета финальной оценки **всегда** берется **последняя** оценка из Grader.

5. Правила оформления задания

Перед отправкой задания оставьте, пожалуйста, отзыв о домашнем задании по ссылке:https://rebrand.ly/bdmade2022q2_feedback_hw. Это позволит нам скорректировать учебную нагрузку по следующим заданиям (в зависимости от того, сколько часов уходит на решение ДЗ), а также ответить на интересующие вопросы.

Оформление задания:

- Код задания (Short name): **HW05:Spark_RDD**.
- Выполненное ДЗ запакуйте в архив **BD-MADE-2022-Q2**_<Surname>_<Name>_HW#**.zip**, например, для Алексея Драля **BD-MADE-2022-Q2**_Dral_Alexey_**HW05**.zip. Если ваше решение лежит в папке my_solution_folder, то для создания архива hw.zip на Linux и Mac OS выполните команду[5]:
  - zip -r hw.zip my_solution_folder/*
- На Windows 7/8/10: необходимо выделить все содержимое директории `my_solution_folder/` нажать правую кнопку мыши на одном из выделенных объектов, выбрать в открывшемся меню "Отправить >", затем "Сжатая ZIP-папка". Теперь можно переименовать архив.
- Решения заданий должны содержаться в одной папке.
- Перед проверкой убедитесь, что дерево вашего архива выглядит так:
  - **| BD-MADE-2022-Q2_<Surname>_<Name>_HW05.zip**
  - **| ---- task_<Surname>_<Name>_bigram.py**
  - **| ---- task_<Surname>_<Name>_bigram.out**
  - **| ---- task_<Surname>_<Name>_collocation.py**
  - **| ---- task_<Surname>_<Name>_collocation.out**
  - При несовпадении дерева вашего архива с представленным деревом, ваше решение будет невозможно автоматически проверить, а значит, и оценить его.
- Для того, чтобы сдать задание, необходимо:
  - Зарегистрироваться и залогиниться в сервисе Everest
  - Перейти на страницу приложения: MADE BigData Grader

[5] Флаг -r значит, что будет совершен рекурсивный обход по структуре директории

- Выбрать вкладку Submit Job (если отображается иная).
- Выбрать в качестве “Task” значение: **HW05:Spark_RDD**[6]
- Загрузить в качестве “Task solution” файл с решением
- В качестве Access Token указать тот, который был выслан по почте

Любые вопросы / комментарии / предложения можно писать в телеграм-канал курса или на почту bd_made2022q2@bigdatateam.org.

Всем удачи!

[6] Сервисный ID: spark.rdd.onsite_hw

6. FAQ (часто задаваемые вопросы)

"You are not allowed to run this application", что делать?
Если Вы видите надпись "You are not allowed to run this application" во вкладке Submit Job в Everest, то на данный момент сдача закрыта (нет доступных для сдачи домашних заданий, по техническим причинам или другое). Попробуйте, пожалуйста, еще раз через некоторое время. Если Вы еще ни разу не сдавали, у коллег сдача работает, но Вы видите такое сообщение, сообщите нам об этом.

Grader показывает 0 или < 0, а отчет (Grading report) не помогает решить проблему
Ситуации: • система оценивания показывает оценку (Grade) < 0, а отчет (Grading report) не помогает решить проблему. Пример: в случае неправильно указанного access token система вернет -401 и информацию о том, что его нужно поправить; • система показывает 0 и в отчете (Grading report) не указано, какие тесты не пройдены. Пример: вы отправили невалидный архив (rar вместо zip), не приложили нужные файлы (или наоборот приложили лишние - временные файлы от Mac OS и т.п.), рекомендуется проверить содержимое архива в консоли: `unzip -l your_solution.zip` Если Вы столкнулись с какой-то из них присылайте ссылку на выполненное задание (Job) в чат курса. Пример ссылки: https://everest.distcomp.org/jobs/67893456230000abc0123def

Что в отчете Grader означает проверка X ?
Как читать отчет: Для каждого теста • Raw_score - балл за конкретный тест. Может быть как бинарным (1\0), так и находиться в интервале от 0 до 1 • Score - Raw_score*weight (вес теста в общей оценке). Вес указан для каждого теста ниже

Итоговая оценка: смотрите строку Score (сумма Score всех индивидуальных тестов) внизу отчета.

Общие тесты:

- test_unzip_is_succesful - ДЗ заархивировано в .zip архив и грейдер может его разархивировать
- test_py_files_min_lint_score - качество кода в .py файлах оценивается выше 8.0

Тесты spark.bigram:

- test_expected_bigram_count_found - найдено нужное количество биграмм
- test_bigrams_are_sorted - биграммы отсортированы лексикографически
- test_bigrams_has_underscore_between_words - слова в биграмме разделены символом “_”
- test_all_correct_bigrams_are_found - найдены требуемые биграммы и они отсортированы в нужном порядке
- test_expected_format_for_spark_bigram_stdout - формат вывода соответствует шаблону
- test_bigrams_and_counts_are_correctly_calculated - расчет верен

Тесты spark.collocation:

- test_expected_collocation_count_found - найдено нужное количество колллокаций
- test_collocations_are_sorted_desc_by_value - данные отсортированы по убыванию NPMI
- test_collocations_has_underscore_between_words - слова в биграмме разделены символом “_”
- test_all_collocations_are_sorted_correctly - получили все искомые коллокации и они отсортированы в правильном порядке
- test_expected_format_for_spark_collocation_stdout - формат вывода соответствует шаблону
- test_is_npmi_correctly_calculated - значение NPMI рассчитано корректно для каждой коллокации
- test_solution_finish_within_max_execution_time - не превышен порог по времени