MapReduce 2. Вариант 1. Фильтр

Сечко Никита

Извлечение данных из статей википедии

Извлечение данных реализовано в файле `wiki.py` в классе `WikipediaParser` в виде двух статических методов: `get_content` и `get_links`.

Если при попытке получения данных будет получен отличный от 200 код возврата, то появится соответствующая запись в потоке ошибок, но работа скрипта будет продолжена.

- `get_content(url: str)`:
  `url` — строка, являющаяся ссылкой на статью из википедии.

  Название статьи извлекается из блока `<h1 id="firstHeading" class="firstHeading">`, а текст статьи из блока `<div id="mw-content-text">`

  Возвращаемое значение является нормализованной конкатенацией текста из этих блоков.

- `get_links(url: str)`:
  `url` — строка, являющаяся ссылкой на статью из википедии.

  Возвращаемое значение является списком ссылок из данной статьи на другие статьи википедии.

  Ссылки извлекаются из блоков `<a>`. При этом нас интересуют только ссылки на википедию. Также нас не интересуют ссылки на картинки, документы, обсуждения, и любые другие данные, которые не являются статьями из википедии. Поэтому ссылки фильтруются данным регулярным выражением:

```
(?=^/wiki/.+)
(?!^/wiki/User:.*)
(?!^/wiki/File:.*)
(?!^/wiki/Category:.*)
(?!^/wiki/Special:.*)
(?!^/wiki/Wikipedia:.*)
(?!^/wiki/Talk:.*)
(?!^/wiki/Template:.*)
(?!^/wiki/Template_talk:.*)
(?!^/wiki/Media:.*)
(?!^/wiki/User_talk:.*)
(?!^/wiki/Help:.*)
(?!^/wiki/Module:.*)
(?!^/wiki/Free_Content:.*)
(?!^/wiki/Portal:.*)
```

Генерация исходных данных

Скрипт для генерации содержится в файле `generator.py`. Скрипт генерирует два файла: список ссылок на статьи википедии и словарь для фильтрации.

Генерация списка ссылок

Список ссылок генерируется посредством обхода в ширину ссылок на статьи из википедии, начиная с некоторой стартовой статьи. Для получения ссылок из статьи википедии используется метод `get_links` из файла `wiki.py`.

Т.к. самая долгая часть алгоритма — ожидание ответа от сервера, то для ускорения работы используется класс `Thread`. Данный класс на самом деле ничего не распараллеливает в силу `GIL(Global Interpreter Lock)` в `Python`, но в нашем случае он отлично помогает во время ожидания ответа от сервера выполнять другую полезную работу.

Генерация словаря

Для генерации словаря я использовал корпус Брауна из библиотеки `NLTK`. В данном корпусе нет некоторых современных слов, но для генерации тестового набора для нашей задачи хватит.

Чтобы более частые слова имели больший приоритет при генерации, я посчитал количество вхождений каждого слова в корпус и выдал им пропорциональные вероятности появления в списке. Стоп-слова я исключил.

Интерфейс генератора

Запуск генератора осуществляется с помощью команды `python3 generator.py`. Также генератор позволяет настраивать параметры генерации при помощи следующих параметров:

- `-n` — количество ссылок, которое необходимо сгенерировать. Значение по умолчанию — `1000`;
- `-u` — ссылка, с которой необходимо начать генерацию. Значение по умолчанию — `https://wikipedia.org/wiki/Wikipedia`;
- `-d` — файл, в которой необходимо сохранить ссылки. Значение по умолчанию — `urls.txt`;
- `-c` — количество слов, которое необходимо сгенерировать. Значение по умолчанию — `10`;
- `-w` — файл, в которой необходимо сохранить слова. Значение по умолчанию — `words.txt`.

Решение

Для решения задачи используются два скрипта: `wiki_map.py` и `wiki_reduce.py`.

Map стадия

Она реализована в скрипте `wiki_map.py`.

На данной стадии список ссылок разделяется на блоки, каждый из которых обрабатывается скриптом отдельно, а результаты конкатенируются в один файл. Данный скрип по сути решает исходную задачу целиком на небольшом списке ссылок. Работает он следующим образом:

- Слова из словаря считываются и сохраняются в `set`. Слов в языке не так уж и много, например мой генератор насобирал со всего корпуса Брауна около `42000` различных слов. Поэтому мы можем себе позволить при каждом вызове скрипта сохранять их все в оперативную память;

- При помощи метода `get_content` класса `WikipediaParser` из статьи достаётся нормализованный текст. На данном этапе, аналогично генерации, используется класс `Thread` для уменьшения времени ожидания ответа от сервера;

- В полученных текстах статей слова проверяются на принадлежность словарю. Если было обнаружено слово из словаря, то ссылка добавляется во множество ссылок, содержащих данное слово.

Также, несмотря на то, что в качестве разделителя в условии рекомендовалось использовать `#`, я для этих целей использовал пробел, поскольку `#` иногда встречается в ссылках, а пробел — нет.

У данного скрипта есть один параметр:

- `-w` — файл со словарём. Значение по умолчанию — `words.txt`.

Reduce стадия

После Map стадии в файле для каждого слова есть много списков ссылок, которые идут в хаотичном порядке. На Reduce стадии все эти списки для одного и того же слова объединяются в один большой список, а также слова и ссылки сортируются в лексикографическом порядке.

Тестирование

К сожалению, мне не удалось протестировать своё решение на действительно большом списке ссылок, поскольку уже при списке из `10000` ссылок я с некоторого момента начинаю получить от википедии код `429(Too many requests)`. Но на `1000` ссылок программа замечательно отрабатывает.