Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Кафедра автоматизації проектування енергетичних процесів і систем

Звіт

з лабораторної роботи №2

“ЗАТОСУВАННЯ SPARKSQL, РОБОТА З ДАНИМИ З

ВИКОРИСТАННЯМ DATAFRAMES ТА DATASET”

Виконав:

студент 5-го курсу

ТЕФ

гр. ТМ-01мп  
Каліка Б.М.

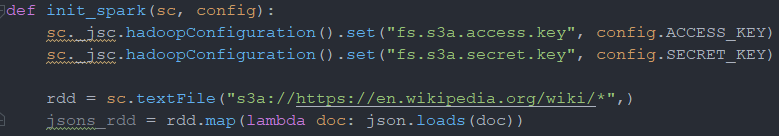
Перевірив:

Проф. Федорова Н.В.

Київ – 2020

**Конфігурація**

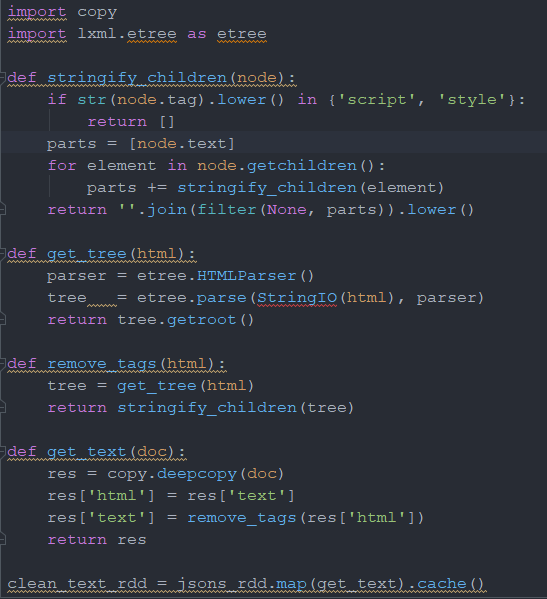
Вказуємо параметри підключення до Spark і створємо RDD з даних і заодно відразу розпарсуємо документи з json-формату:



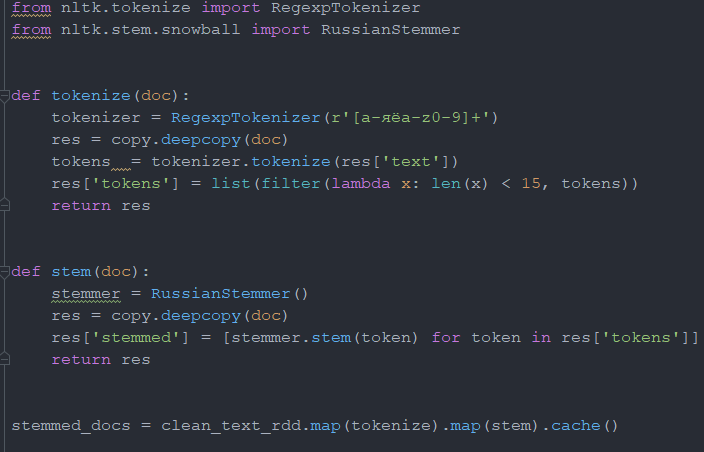
Тут ми застосували одну з базових функцій спарка - map, яка застосовує функцію до всіх елементів масиву, роблячи це паралельно на всіх вузлах кластера. Далі ми кілька разів застосуємо цю функцію для попередньої обробки тексту.

**Очищення тексту від HTML-розмітки**

Для цього використовуємо бібліотеку lxml для парсинга html-а і видалення розмітки:



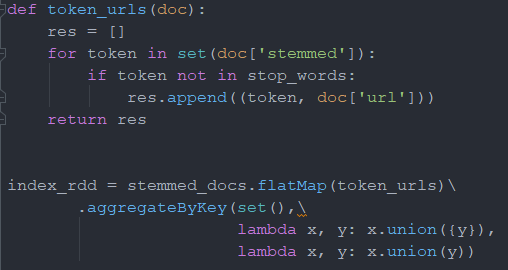
**Використовуємо текенизацію**



Функція cache (), викликана після функції map (), підказує, що цей датасета необхідно закеширувати. Якщо цього не зробити, при багаторазовому використанні спарк буде його розраховувати заново.

**Побудова зворотного індексу**

Завдання побудувати датасета типу слово -> безліч URL дуже схожа на завдання порахувати кількість документів, в яких зустрічається слово, з однією відмінністю: ми будемо не додавати одиницю кожного разу, а додавати новий URL в безліч.



Тут на додаток до вже використаної раніше функції flatMap використовується ще і функція aggregateByKey, яка дуже схожа на reduceByKey, але приймає три параметри.

- порожній об'єкт-акумулятор, в якому буде накопичуватися результат;

- функцію, яка додає в акумулятор одне значення;

- функцію, яка може злити два акумулятора в один. Значення для одного ключа можуть агрегуватися паралельно, ця функція потрібна для об'єднання частково агрегованих результатів.

Далі залишилося тільки зберегти зворотний індекс. Для того щоб його зберегти, нам підійде будь-який розподілене key-value сховище. Я вибрав aerospike- він швидкий, добре розподіляється. Записувати в значення для токена прямо серіалізовані безліч url-ів.

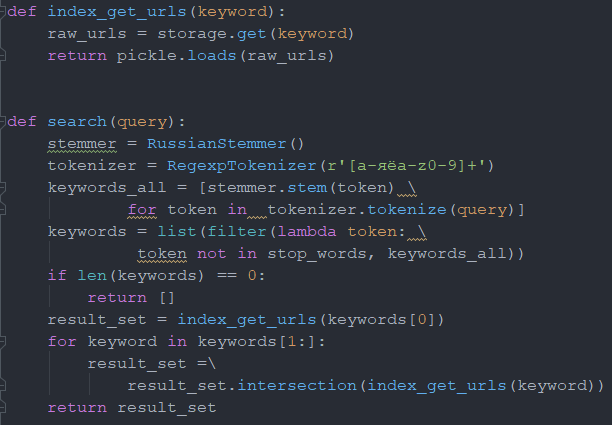
**Сериализация данних**

Для цього використовую модуль pickle - стандартний пітоновскій спосіб сериализации майже будь-яких об'єктів. Також використовую невелику обгортку над стандартним клієнтом aerospike, яка дозволяє форматувати з'єднання з базою даних в момент першого запису або читання. Це потрібно, так як spark не може распараллелить підключення до бази даних по всіх вузлах кластера, доводиться кожен раз підключатися заново.



**API для отримання даних**

Реалізовуємо функцію, яка буде виконуватися під час користувальницького запиту. З нею все просто: розбиваємо запит на маркери, витягаємо безлічі URL-ів для кожного токена і перетинаємо їх:



Запускаємо.