University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**Big Data H / Big Data M**			
Coursework Number	**1**			
Deadline	**Time:**	**23:59**	**Date:**	**Sun 6 Mar 2021**
% Contribution to final course mark	**20% (H) / 20% (M)**			
Solo or Group ✓	**Solo**		**Group**	✓
Anticipated Hours	**8 per group member**			
Submission Instructions	**Submit via Moodle one pdf file containing your report, plus one uog-bigdata.zip file as per detailed instructions**			
Please Note: This Coursework cannot be Re-Done				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via
https://webapps.dcs.gla.ac.uk/ETHICS for all coursework
UNLESS submitted via Moodle

Big Data (H/M) Assessed Exercise Task Sheet

Summary

The goal of this exercise is to familiarize yourselves with the design, implementation and performance testing of Big Data analysis tasks using Apache Spark. You will be required to design and implement a single reasonably complex Spark application. You will then test the running of this application locally on a small data sample, and remotely on a pre-prepared Spark cluster over the full dataset. Finally, you will write a short report describing your design, design decisions, and where appropriate critique your design. You will be evaluated based on code functionality (does it produce the expected outcome), code quality (is it well designed and follows good software engineering practices) and efficiency (how fast is it and does it use resources efficiently), as well as your submitted report.

Task Description

You are to develop a batch-based text search and filtering pipeline in Apache Spark. The core goal of this pipeline is to take in a large set of text documents and a set of user defined queries, then for each query, rank the text documents by relevance for that query, as well as filter out any overly similar documents in the final ranking. The top 10 documents for each query should be returned as output. Each document and query should be processed to remove stopwords (words with little discriminative value, e.g. 'the') and apply stemming (which converts each word into its 'stem', a shorter version that helps with term mismatch between documents and queries). Documents should be scored using the DPH ranking model. As a final stage, the ranking of documents for each query should be analysed to remove unneeded redundancy (near duplicate documents), if any pairs of documents are found where their titles have a textual distance (using a comparison function provided) less than 0.5 then you should only keep the most relevant of them (based on the DPH score). Note that there should be 10 documents returned for each query, even after redundancy filtering.

You will be provided with a Java template project like the tutorials already provided. Your role is to implement the necessary Spark functions to get from a Dataset<NewsArticle> (the input documents) and a Dataset<Query> (the queries to rank for) to a List<DocumentRanking> (a ranking of 10 documents for each query). Your solution should only include spark functions, apart from any final processing you choose to do within the driver program. You should not perform any 'offline' computation (e.g. pre-constructing a search index), i.e. all processing should happen during the lifecycle of the Spark app. The template project provides implementations of the following code to help you:

- Loading of the query set and converting it to a Dataset<Query>.
- Loading of the news articles and converting it to a Dataset<NewsArticle>
- A static text pre-processor function that converts a piece of text to its tokenised, stopword removed and stemmed form. This function takes in a String (the input text) and outputs a List<String> (the remaining terms from the input after tokenization, stemming and stopword removal).

- A static DPH scoring function that calculates a score for a <document,term> pair given the following information:
    - Term Frequency (count) of the term in the document
    - The length of the document (in terms)
    - The average document length in the corpus (in terms)
    - The total number of documents in the corpus
    - The sum of term frequencies for the term across all documents
- A static string distance function that takes two strings and calculates a distance value between them (within a 0-1 range).

The DPH score for a <document,query> pair is the average of the DPH scores for each <document,term> pair (for each term in the query). When designing your solution, you should primarily be thinking about how you can efficiently calculate the statistics needed to score each document for each query using DPH.

Dataset

The dataset that you will be using for this exercise is a corpus of news articles from the Washington Post, along with a set of queries. There are two versions of this dataset: the local sample; and the full dataset. The local sample is there to allow you to quickly iterate on your design using local spark deployments (like in the tutorials). This will be provided as part of the template project and contains 5,000 news articles and three queries. The full dataset is comprised of around 670,000 documents and 10 queries. A query contains the following information:

```
String originalQuery;     // The original query unaltered
List<String> queryTerms; // Query terms after tokenization, stopword
                            removal and stemming
short[] queryTermCounts; // The number of times each query term appears
```

The pre-provided code that constructs the Dataset<Query> already performs tokenization, stopword removal and stemming on the queries, the outcome of which is stored in the queryTerms variable.

A news article contains the following information:

```
String id; // unique article identifier
String article_url; // url pointing to the online article
String title; // article title
String author; // article author
long published_date; // publication date as a unix timestamp (ms)
List<ContentItem> contents; // the contents of the article body
String type; // type of the article
String source; // news provider
```

For this exercise, you only need to use the ‘id’, ‘title’ and ‘contents’ fields. Below is an extract from an example news article. As you can see, the ‘contents’ of a news article is a list of elements from within that news article, such as the kicker line, title element, header image, and various paragraphs. When calculating DPH for a document, you should only consider terms within the title and within ContentItem elements that have a non-null subtype and that subtype is listed as “paragraph”. Additionally, if an article has more than 5 paragraphs, you need only consider the first 5. No text pre-processing is applied by the provided code that constructs the Dataset<NewsArticle>.

```
JSON
  id : "b2e89334-33f9-11e1-825f-dabc29fd7071"
  article_url : "https://www.washingtonpost.com/sports/colleges/danny-coale-jarrett-boykin-are-a-perfect-1-2-punch-for-virginia-tech/2011/12/31/gIQAAaW4SP_story.html"
  title : "Danny Coale, Jarrett Boykin are a perfect 1-2 punch for Virginia Tech"
  author : "Mark Giannotto"
  published_date : 1325376562000
  contents
    0
      content : "Colleges"
      mime : "text/plain"
      type : "kicker"
    1
      content : "Danny Coale, Jarrett Boykin are a perfect 1-2 punch for Virginia Tech"
      mime : "text/plain"
      type : "title"
    2
      fullcaption : "Virginia Tech wide receiver Danny Coale (19) was lightly recruited out of Episcopal in Alexandria but has teamed with Jarrett Boykin for a record-breaking receiving duo. (John Bazemore/AP)"
      imageURL : "https://img.washingtonpost.com/rw/2010-2019/WashingtonPost/2011/11/12/Sports/Images/Virginia_Tech_Georgia_Tech_Football_0cf3f.jpg"
      mime : "image/jpeg"
      imageHeight : 2260
      imageWidth : 2775
      type : "image"
      blurb : "Virginia Tech wide receiver Danny Coale (19) was lightly recruited out of Episcopal in Alexandria but has teamed with Jarrett Boykin for a record-breaking receiving duo. (John Bazemore/AP)"
    3
      content : "By Mark Giannotto"
      mime : "text/plain"
      type : "byline"
    4
      content : 1325376562000
      mime : "text/plain"
      type : "date"
    5
      content : "NEW ORLEANS — Whenever a Virginia Tech offensive coach is asked how the most prolific receiving duo in school history came to be, inevitably the first road game in 2008 against North Carolina comes up."
      subtype : "paragraph"
      type : "sanitized_html"
      mime : "text/html"
    6
      content : "Midway through the first quarter, Virginia Tech had to call two timeouts in a row because then-freshmen Jarrett Boykin and Danny Coale couldn't seem to line up right, and "they had those big eyes out there looking around,
      subtype : "paragraph"
      type : "sanitized_html"
      mime : "text/html"
```


Figure 1: Example News Article

Submitted Report

In addition to your spark code-base, you are also asked to submit a short report describing the design logic for your solution. This should be around 2 A4 pages in length total. You should include a section for each of the Spark functions you developed where for each you summarize:

- What the goal of that function is
- How it fits into your overall pipeline

Then, in a final section, you should include a paragraph discussing why you think that your solution is efficient given the requirements of the project, highlighting any decisions you took with the aim of improving its efficiency. You may wish to also highlight any challenges you faced in the implementation and how you overcame those challenges.

Code Management

For this project you are provided a git repository for your project at https://stgit.dcs.gla.ac.uk. You should have already received an email with the details on how to access this repository. For this project you are required to commit code you work on week-by-week to this repository. This repository has two main uses:

1. Our remote deployment service uses this repository to get your source code for remote testing (more on this in weeks 5-6)
2. We use this as a source of evidence that each team member is contributing to the project

Make sure that each member can access the repository and that their local machine is configured to commit content using their student account (so we can trace it to them).

What to hand in

You should submit via Moodle:

- A single pdf file that containing your final report.
- A copy of your code-base as a single zip file, downloaded from your team's git repository.

How this exercise will be marked

Following timely submission on Moodle, the exercise will be given a numerical mark between 0 (no submission) and 20 (perfect in every way). The numerical marks will then be converted to a band (A5, A4, etc.). The marking scheme is as follows:

- 12 marks for correct implementation (partial marks for partially correct implementation).
- 2 marks for quality/readability of your source code.
- 2 marks for documentation of your source code.
- 4 marks for efficiency/scalability of your source code.