BigData Assessed Exercise Report

Program Logic Summary

As described in the task description, this task focused on deriving the top ten non-similar news articles in terms of DPH scores by query items. In the first step, the data is processed to obtain the news dataset according to the functions provided, and it is ensured that each piece of data includes at most five paragraphs of content and is split into items. The second step is to create a new data type, using a HashMap to associate the word frequencies of the query words with each article. The third step is to Iterate through each query item, calculate the DPH score for each term in a query to get the average DPH score of the query words, and then store the results in RankedResult. Finally, the RankedResult of each query is sorted according to the descending order of DPH scores, and then the similarity of articles is calculated according to TextDistanceCalculator to remove similar articles and return the top ten documents in DPH ranking.

Our final code stored in main branch.

Custom Functions

NewsFormaterMap

Make the title and contents of the news be processed by stopword and stemming, and the contents only contain the contents of the first five paragraphs in the original data (the words of each piece of content after processing are separated by commas), which is convenient for later processing. This step can get the total number of articles.

DocumentLengthMap

Calculate the length of each article, get a Dataset <Integer> after map operation to store the length of each article.

IntSumReducer

This summation reducer is used to calculate the total length of the corpus articles using the results stored in the Dataset<Integer>, and divide it with the total number of articles obtained before to obtain the average length of the articles.

TfDocument

A TfDocument class instance includes three elements, a HashMap(the key-value pair is a single query word) and the corresponding article word frequency (String, Short), the corresponding NewsArticle and NewsArticle article length

TfFilterFlatMap

Map the previous news to TfDocument. This function is used twice. The first time, the passed parameters are used as all query words to get the corresponding frequency of each query word. The second time, it is called in traversing each query to get the HashMap for a query and its corresponding article in order to calculate the DPHScore for each query in each loop.

CorpusTfReducer

Merge the HashMap of two different TfDocument class instances to calculate the total word frequency of each query term.

QueryResultFlatMap

Calculate the DPHScore for each set of queries, where the DPHScore of the query terms in the query is calculated one by one, and then the results are averaged to obtain the DPHScore of the query, and the results are stored in the RankedResult instance

Top10Result:

This function sorts the obtained RankedResult in descending order, to ensure that the top10 is obtained and the article with the higher HPD score is retained among the two similar articles. A new result set is created, the TextDistance is calculated by traversing the previous RankedResult, comparing the articles in the result set with the articles in the previous RankedResult, and adding new articles to the result set when the similarity score between the two articles is less than 0.5. When the number of articles in the result set reaches ten, the result set is terminated, and the result set is stored in the DocumentRanking instance. Here we keep those article with a null title, we think due to the DPHScore we'd better keep it.

Efficiency Discussion

- For data reading, sometimes reading very large files, the size of each file part is very large, if read directly, spark will read a part file in many tasks according to the file size, the default is 128M, so that there will be tens of thousands or even hundreds of thousands of tasks working in the reading phase, you can set parameters to customize. This can be considered for subsequent optimization in the project.
- We use the broadcast variable, the number of words (bcallFrequencyinCorpus) and a hash map of the data terms collected in the documents (CorpusTfReducer). Because we use the same hash map for all queries to find the value to calculate the DPH score, the number of copies of the variable is greatly reduced, thus reducing the overhead on the Executor's memory footprint.

Challenges

- In terms of data handling, the TfDocument type originally only included HashMap and news article, as the article length was previously stored in Dataset<Integer> DocumentLength. However, subsequent calculations require a mapping between article length and article, so the int DocLength is added to the structure of TfDocument later.
- The biggest challenge for us was the timeline, which was so tight due to a mis-assessment of the difficulty of the task that we didn't finish the project until the last few days.