[Recitation 6]

Paahuni Khandelwal
Email: cs435@cs.colostate.edu

20th September, 2019
Submission

- Submission Deadline for PA2 - 24th October (by 5 pm)
- Worth 10 points (10% of your grades)
- Dataset - same as for PA1
How to design Jobs for PA2

- There are few documents (articles) without article body. (For example: HelloWorld<====>12345<====> )
- Put a checker before you proceed. Also, ignore such articles while calculating N (total number of articles in the corpus).
- Divide the PA2 into two parts (A and B). From Part A, get the TF-IDF value for each unigram in each document.
- From Part B, you should get article summary for any given article (provided during demo).
- Use multiple reducers wherever possible and implement CustomPartitioner
- Make sure Part A and Part B are separate jars that could be run independently.
- To calculate N and pass its value to a different job, use Hadoop's **Counters** class.

  - References:

    https://hadoop.apache.org/docs/r2.7.4/api/org/apache/hadoop/mapred/Counters.html

Preprocess unigrams ie remove non-alphanumeric characters

Find the frequency of each unigram in each article.

Similar to Profile2 in PA1

Output will be in the form: <docID, (unigram frequency)>
• Calculate TF of each unigram i in each article j using:
  \[ TF_{ij} = 0.5 + 0.5 \left( \frac{f_{ij}}{\max_k f_{kj}} \right) \]

  where, \( \max_k f_{kj} \) is the highest frequency of a unigram in the article j.

• Mapper output will be in the form: \(<K,V> \rightarrow <docID, \text{(unigram frequency)}>\)

• Reducer:
  
  – First iterate through all the values and find \( \max_k f_{kj} \)
  
  – Next, calculate TF for each unigram
  
  – Output will be in the form: \(<docID, \text{(unigram TF value)}>\)
• In this job, you can calculate total number of documents in entire corpus

• Calculate $n_i$, total number of documents in the whole corpus, where unigram $i$ occurred at least once

• Mapper output will be in the form: $<K,V> -> <\text{unigram}, (\text{docID unigram TFvalue})>$

• Reducer:
  - In setup() : Fetch total number of documents counter value from previous map job
  - Iterate through all the input values and find $n_i$.
  - Calculate idf value as well, finally push tf * idf as one of the outputs
  - Output will be in the form: $<\text{docID}, (\text{unigram TfIdf})>$
Profile A Job 3 (Continued)

- Calculate **TF-IDF** of each unigram \( i \) in each article \( j \) using: \( \text{IDF}_i = \log_{10}(N/n_i) \)

- \( \text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_{ij} \)

- Calculate \( \text{IDF} \) for each unigram.

- Calculate TF-IDF for each unigram.

- Output will be in the form: \(<\text{docID}, \text{(unigram TF-IDF value)}>\)

**Note:** To calculate \( \text{IDF} \), you need to pass \( N \) that you calculate in previous job.
Profile B Job 1

- Generate summary for any given article.
- Split by a period followed by single space ( . ) to fetch sentences
- Then only preprocess unigrams ie removing non-alphanumeric characters
- Multiple inputs required: Output of previous job (PartA-Job4) and given article. You can implement this in either of the two ways:
  - Use `MultipleInputs.addInputPath()` and join them on keys. (Reduce Side Join)
  - Persist output from previous job in Hadoop’s DistributedCache.