CS435: Introduction to Big Data

GTA: Bibek R. Shrestha
Email: cs435@cs.colostate.edu

March 09, 2018
Today...

- Discussion on Programming Assignment 2
Few changes not covered in last recitation

- First, calculate TF-IDF values of each word in each of the document (represented by unique DocumentID, as in PA1).
- Then, for each sentence in a document, identify top 5 words with highest TF-IDF values. (If there are less than 5 words in a sentence, use all the words.).
- Sum TF-IDF values of these 5 words. This is TF-IDF value for this sentence, $\text{Sentence}_{\text{TF-IDF}}$.
- Calculate $\text{Sentence}_{\text{TF-IDF}}$ for each sentence in the document.
- Select top 3 sentences with highest $\text{Sentence}_{\text{TF-IDF}}$.
- These three sentences form a summarization for the document.
- Perform above steps to get summarization for each document in the complete dataset.
How to program PA2: Part-A?

First MapReduce Job:

- Similar to first part of Profile 2 in PA1
- Calculate frequency of word in each document.
- Output of Mapper will be in the form:
  - (DocumentID, unigram, 1)
- Output from this job/reducer will be in the form:
  - (DocumentID, unigram, frequency)
- You can use CompositeKey class or you can just concatenate your keys as single string with a delimiter in-between.
Second MapReduce Job:

- Calculate TF value
- Use output from your first mapreduce job.
- Output of Mapper will be in the form:
  - (DocumentID, {unigram, frequency})
- In your reducer, you will have a list of {unigram, frequency} (as value) for each DocumentID (as key).
- Find the max. frequency $\max_k f_{kj}$
- Calculate TF for each unigram using:
  - $[ TF_{ij} = 0.5 + 0.5 ( f_{ij} / \max_k f_{kj} ) ]$
- Output of Reducer will be in the form:
  - (DocumentID, {unigram, frequency, TFvalue})
Third MapReduce Job:

- Calculate IDF value
- For that, first calculate $n_i$, total number of documents in whole corpus where unigram $i$ occurred, for: $\text{IDF}_i = \log_{10}(N/n_i)$
- Use output from your second mapreduce job.
- Output of Mapper will be in the form:
  - (unigram, \{DocumentID, TFvalue\})
- In your reducer, you will have a list of \{DocumentID, TFvalue\} (as value) for each unigram (as key).
- The size of the list gives $n_i$.
- Output of Reducer will be in the form:
  - (unigram, \{DocumentID, TFvalue, $n_i$\})
Break...How to get the value of $N$?

- $N$ is the total number of unique documents in the given dataset.
- Used to calculate IDF as: $IDF_i = \log_{10}(N/n_i)$
- We can use Counters class implemented in Hadoop.
- Do not hard-code this value.
- References:
  
  
  https://stackoverflow.com/questions/27325536/how-to-access-hadoop-counters-values-via-api
Fourth MapReduce Job:

- Calculate IDF and TF-IDF values
- With output from previous job, \((\text{unigram}, \{\text{DocumentID}, \text{TFvalue}, n_i\})\), calculate IDF using: \(\text{IDF}_i = \log_{10}(N/n_i)\)
- Using TF and IDF, calculate TF-IDF.
- Output of Mapper will be in the form:
  - \((\text{DocumentID}, \{\text{unigram}, \text{TFvalue}, \text{TF-IDFvalue}\})\)
- In reducer, arrange your output field, if required.
- Functionally, reducer is not required. Identity Reducer!
How to program PA2: Part-B?

Fifth MapReduce Job:

- Calculate $\text{Sentence}_{TF-IDF}$ values
- We have to use output from previous job and the data from original dataset. Why?
  - Hint: Use MultipleInputs.addInputPath() and join them on keys. OR, use output of fourth job as lookup. May be persist it in Hadoop’s DistributedCache!
- Split on periods to get each sentence. Tokenize to get each unigram.
- Now use the output from fourth job to get TF-IDF value for each unigram.
- Select top 5 words with highest TF-IDF values. Sum those values. This is $\text{Sentence}_{TF-IDF}$.
- Calculate $\text{Sentence}_{TF-IDF}$ for each sentence in the document.
- Mapper output will be in the form:
  - (DocumentID, {eachSentence, $\text{Sentence}_{TF-IDF}$})
- In your reducer, select top 3 sentences with highest $\text{Sentence}_{TF-IDF}$.
- Final output will be in the form:
  - (DocumentID, (Top three sentences))
In the next recitation...

- Introduction to Programming Assignment 3