Lab Lecture #3

Introduction

Suppose we have a set of English text documents and wish to determine which document is most relevant to the query "*the brown cow*". A simple way to start out is by eliminating documents that do not contain all three words "*the*", "*brown*" and "*cow*", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document and sum them all together; the number of times a term occurs in a document is called its **term frequency**. However, because the term "*the*" is so common, this will tend to incorrectly emphasize documents which happen to use the word "*the*" more, without giving enough weight to the more meaningful terms "*brown*" and "*cow*". Also the term "*the*" is not a good keyword to distinguish relevant and non-relevant documents and terms like "*brown*" and "*cow*" that occur rarely are good keywords to distinguish relevant documents from the non-relevant documents. Hence an **inverse document frequency** factor is incorporated which diminishes the weight of terms that occur very frequently in the collection and increases the weight of terms that occur rarely.

The *term count* in the given document is simply the number of times a given term appears in that document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document d_j. Thus we have the **term frequency**, defined as follows.

$$tf_{ij} = \frac{n_{ij}}{N_j}$$

where n_{ij} is the number of occurrences of the considered term (t_i) in document d_j, and the denominator N_j is the sum of number of occurrences of all terms in document d_j (the document length).

The **inverse document frequency** is a measure of the general importance of the term (obtained by dividing the total number of document by the number of documents containing the term, and then taking the logarithm of that quotient).

$$idf_i = \log \frac{|D|}{M_i}$$

where:

- $|D|$ is total number of documents in the corpus;
- $M_i = |\{d : t_i \in d\}|$ is the number of documents where the term t_i appears in.

Then

$$\left(tf - idf\right)_{ij} = tf_{ij} \cdot idf_i$$

A high weight in *tf–idf* is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. The *tf-idf* value for a term will always be greater than or equal to zero.

Consider a document containing 100 words wherein the word "*cow*" appears 3 times. Following the previously defined formulas, the term frequency (TF) for "*cow*" is then 0.03. Now, assume we have 10 million documents and "*cow*" appears in one thousand of these. Then, the inverse document frequency is calculated as ln(10000000/1000) = 9.21. The *tf-idf* score is the product of these quantities: 0.03 × 9.21 = 0.28.

Compute TF-IDF using MapReduce

Given a small collection of documents, we are going to implement *tf–idf* scores using Hadoop. We will need the following information:

- number of times term t_i appears in a given document (n)
- number of terms in each document (N)
- number of documents term t_i appears in (m)
- total number of documents ($|D|$)

We use multiple rounds of Map/Reduce to gradually compute *tf–idf*:

1. **Word frequency in document**: starting from a directory containing a set of text files, we will produce another set of files associating the couple (t_i, d_j) to the number n of times the term t_i appears in the document d_j.

   Mapper: input: (doc name, doc contents)
   output: ((word, doc name), 1)
   Reducer: sums counts for word in document
   outputs ((word, doc name), n)

2. **Word count in document**: starting from the directory containing the output of the previous job, we will produce another set of files associating the couple (t_i, d_j) to the couple (n, N), where N represents the total number of terms in the document d_j.

   Mapper: input: ((word, doc name), n)
   output: (doc name, (word, n))
   Reducer: sums frequencies n for individual terms in the same document
   outputs ((word, doc name), (n, N))

3. **Word frequency in collection**: starting from the directory containing the output of the previous job, we will produce another set of files associating the couple (t_i, d_j) to the triple (n, N, m), where m represents the number of documents the term t_i appears in the document d_j.

   Mapper: input: ((word, doc name), (n, N))
   output: (word, (doc name, n, N))
   Reducer: sums counts for word in collection
   output: ((word, doc name), (n, N, m))

4. **Calculate *tf-idf***: starting from the directory containing the output of the previous job, and assuming $|D|$ is known, we will produce another set of files associating the couple (t_i, d_j) its *tf-idf* score.

   Mapper: input: ((word, doc name), (n, N, m))
   output: ((word, doc name), *tf-idf*)
   Reducer: do nothing
   output: ((word, doc name), *tf-idf*)