TFIDF (Term Frequency-inverse document frequency) is a statistic that measures how import a word is in a document.

In order to compare word occurences in documents of different lengths, we have to use the frequency of the word occurrence to see how the word is prevalent in the document treated now.

The inverse term frequcency part means that the weight of terms which occur a lot diminishes and wthe weight of terms occurring rarely increases.

MAP and reduce implementation:

5000-8.txt can be used as document.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

Section mapper reducer

1. Calculate the number of occurences of the word in document

In the mapper1, we use regex (get rid of stopwords and non-words) to match words and write <word#DocumentName,1> pairs to the stdOut, which the reducer will take as input.

The reducer1 will aggregate those numbers and output the number of words per documents. The function is designed as follows:

Map()：

Input: <documentLineNumer, contents>

Output: <<word#documentName>, 1>

Reduce()：

Input: <<word#documentName>, 1>

Output: <<word#documentName>, n>

1. Calculate the total number of words of each document

We reorder the input in the mapper, by making the key the documentName and the value word and count: <key, value> output is <documentName,word#n>.

The reducer will count the number of words per documentName, and output th word, filename, count and number of words in document. The function is designed as follows:

Map()：

Input: <<word#documentName>, n>

Output: <documentName, <word#n>>

Reduce()：

Input: <documentName, <word#n>>

Output: <<word#documentName>, <n#N>>

2 possible ways now: C.a (Calculate the TF-IDF by adding an extra Mapper) and C.b (Calculate the TF-IDF by including the calculation the the Reducer.

1. Calculate TF-IDF

Map()：

Input: <<word#documentName>, <n#N>>

Output: <word, <documentNamed#n#N#1>>

Reduce()：

Input: <word, <documentNamed#n#N#1>>

Output: <<word#documentName>, <tfidf >>

Interesting variant to put: calculate the tf-idf in a new mapper, or include it in the last reducer? -> see the runtime difference

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \

-input /user/hadoop/wc/input \

-output /user/hadoop/wc/output \

-file /home/hadoop/mapper.py \

-mapper /home/hadoop/mapper.py \

-file /home/hadoop/reducer.py \

-reducer /home/hadoop/reducer.py

Try to: either do it from local and change the code

Or upload them in dropbox

<https://github.com/devangpatel01/TF-IDF-implementation-using-map-reduce-Hadoop-python-/blob/master/reducer2.py>

<https://github.com/devangpatel01/TF-IDF-implementation-using-map-reduce-Hadoop-python-/blob/master/mapper4.py>

import all docs using wget, all docs in dropbox, remove the last ?=0 in the url in order not to get weird names