2nd Assignment

Due: 13 March 2017

Pre-processing the input (10)

In this assignment, you will use the document corpus of pg100.txt (from http://www.gutenberg.org/cache/epub/100/pg100.txt), as in your previous assignments, assuming that each line represents a distinct document (treat the line number as a document id). Implement a pre-processing job in which you will:

- (2) Remove all stopwords (you can use the stopwords file of your previous assignment), special characters (keep only [a-z],[A-Z] and [0-9]) and keep each unique word only once per line. Don't keep empty lines.
- (1) Store on HDFS the number of output records (i.e., total lines).
- (7) Order the tokens of each line in ascending order of global frequency.

Output the remaining lines. Store them on HDFS in TextOutputFormat in any order.

Set-similarity joins (90)

You are asked to efficiently identify all pairs of documents (d_1, d_2) that are similar ($sim(d_1, d_2) >= t$), given a similarity function *sim* and a similarity threshold *t*. Specifically, assume that:

- each output line of the pre-processing job is a unique document (line number is the document id),
- documents are represented as sets of words,
- $sim(d_1, d_2) = Jaccard(d_1, d_2) = |d_1 \cap d_2| / |d_1 \cup d_2|$,
- $t = 0.8$.

Example:

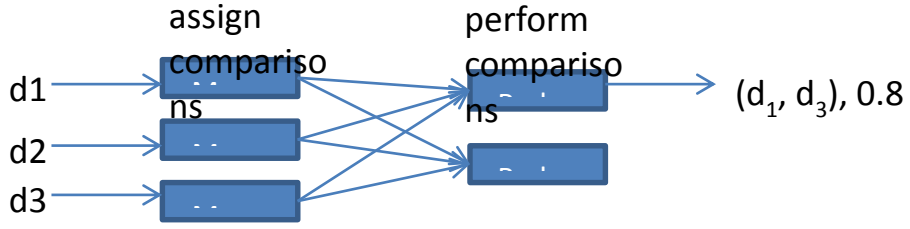
Input:

d_1: "I have a dog"
d_2: "I have a cat"
d_3: "I have a big dog"

$sim(d_1, d_2) = 3/5 < 0.8 \rightarrow$ d1 and d2 are **not** similar
$sim(d_2, d_3) = 3/6 < 0.8 \rightarrow$ d2 and d3 are **not** similar
$sim(d_1, d_3) = 4/5 = 0.8 \rightarrow$ d1 and d3 are similar

Output:

(d_1, d_3), 0.8



Approaches

A naïve approach is to perform all pairwise comparisons between documents and then output only the pairs that are similar. Another approach is to group together documents that have at least one common word and then perform comparisons only between those pairs. Remember the inverted index from the previous assignment... A third approach is to perform indexing, skipping some of the words of each document. For example, when the similarity threshold is 1 (i.e., two documents are considered similar, only if they match 100%), then indexing only a single word for each document would be enough, to skip comparisons between some non-similar documents and guarantee that all similar documents will be compared. Likewise, [Chaudhuri et al. 2006] proves that it is adequate to index only the first $|d| - \lceil t \cdot |d| \rceil + 1$ words of each document d, without missing any similar documents (known as the *prefix-filtering principle*), where t is the Jaccard similarity threshold and $|d|$ is the number of words in d. You are asked to implement the first and the last approach and report your conclusions, along with the execution times and the number of performed comparisons. Specifically:

- **a)** (40) Perform all pair-wise comparisons between documents, using the following technique: Each document is handled by a single mapper (remember that lines are used to represent documents in this assignment). The map method should emit, for each document, the document id along with one other document id as a key (one such pair for each other document in the corpus) and the document's content as a value. In the reduce phase, perform the Jaccard computations for all/some selected pairs. Output only similar pairs on HDFS, in TextOutputFormat.
  Make sure that the same pair of documents is compared no more than once. Report the execution time and the number of performed comparisons.
- **b)** (40) Create an inverted index, only for the first $|d| - \lceil t \ |d| \rceil + 1$ words of each document d (remember that they are stored in ascending order of frequency). In your reducer, compute the similarity of the document pairs. Output only similar pairs on HDFS, in TextOutputFormat. Report the execution time and the number of performed comparisons.
- **c)** (10) Explain and justify the difference between a) and b) in the number of performed comparisons, as well as their difference in execution time.

Related Work:

S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA, page 5, 2006.