Algorithms for Massive Datasets

Course held by Prof. Dario Malchiodi

Federico Cristiano Bruzzone
Id. Number: 27427A

MSc in Computer Science

UNIVERSITY OF MILAN
Computer Science Department
ADAPT-Lab

COMPUTER SCIENCE DEPARTMENT • ADAPT-LAB

Academic Year 2022–2023

Contents

1 HDFS and MapReduce

1.1 Things Useful to Know

1. The **TF.IDF** (*Term Frequency times Inverse Document Frequency*) measure of word importance.
2. Hash functions and their use.
3. Secondary storage (disk) its effect on running time of algorithms.
4. The base e of natural logarithm and identities involving that constant.
5. Power laws.

1.1.1 Importance of Words in Documents

Classification often starts by looking at documents, and finding the significant words in those documents. Our first guess might be that the words appearing most frequently in a document are the most significant. However, that intuition is exactly opposite of the truth. The formal measure of how concentrated into relatively few documents are the occurrences of a given word is called **TF.IDF**. It is normally computed as follows. Suppose we have a collection of N documents. Define f_{ij} to be the *frequency* of term (word) i in document j. Then, define the *term frequency* $\boldsymbol{TF}_{ij}$ to be:

$$\boldsymbol{TF}_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

That is, the term frequency of term i in document j is f_{ij} normalized by dividing it the maximum number of occurrences of any term (perhaps excluding stop words) in the same document.

The IDF for a term is defined as follows. Suppose term i appears in n_i of the N documents in the collection. Then $\boldsymbol{IDF}_i = \log_2(N/n_i)$. The **TF.IDF** score for term i in document j is then $\boldsymbol{TF}_{ij} \times \boldsymbol{IDF}_i$.

1.1.2 Hash Functions

A hash function h takes a *hash-key* value as an argument and produces a *bucket number* as a result. The bucket number is an integer, normally in the range 0 to $B-1$, where B is the number of buckets. Hash-keys can be of any type. There is an intuitive property of hash functions that they “randomize” hash-keys.

1.1.3 Indexes

An *index* is a data structure that makes it efficient to retrieve objects given the value of one or more elements of those objects. The most common situation is one where the objects are records, and the index is on one of the fields of that record. Given a value v for that field, the index lets us retrieve all the records with value v in that field.

1.1.4 Secondary Storage

Disks are organized into *blocks*, which are the minimum units that the operating system uses to move data between main memory and disk. It takes approximately ten milliseconds to *access* and read a disk block. That delay is at least five orders of magnitude (a factor of 10^5) slower than the time taken to read a word from main memory. You can assume that a disk cannot transfer data to main memory at more than a hundred million bytes per second (100MB), no matter how that data is organized. That is not a problem when your dataset is a megabyte. But a dataset of a hundred gigabytes or a terabyte presents problems just accessing it, let alone doing anything useful with it.

1.1.5 The Base of Natural Logarithms

The constant $e = 2.7182818...$ has a number of useful special properties. In particular:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$$

Bibliography