Reporte Técnico, No. 1, 2017
Universidad EAFIT

Clústering de Documentos a partir de Métricas de Similitud

Mauricio Hoyos[1], Jonathan Zapata[2]

Abstract
Text mining is an analysis technique which has allowed us to implement a set of new applications through the time. Such as search engines in the web (Google, Facebook, Amazon, Spotify, Netflix, among others), suggestions systems, natural language processing and others. The document clustering techniques enable us to link a document with other similar documents according to a comparison metric. The basic idea of the proposed implementations is to compare the efficiency between computing in a single node and computing in a distributed network of nodes.

Keywords
K-means — TF-IDF — Apache Spark — Mlib — Cluster — Bigdata — Distributed Computing — Distributed Storage — Similarity

[1] *Departamento de Ingeniería de Sistemas, Universidad EAFIT, Medellín, Colombia,* **mhoyosa2@eafit.edu.co**
[2] *Departamento de Ingeniería de Sistemas, Universidad EAFIT, Medellín, Colombia,* **jzapat80@eafit.edu.co**

Contents

	Introducción	**1**
1	**Marco Teórico**	**1**
2	**Implementación**	**2**
3	**Análisis de resultados (secuencial vs. paralelo)**	**2**
	References	**2**

Introducción

Actualmente, debido a la gran cantidad de información que se encuentra en los medios, y a que está alojada en diferentes bases de datos, surge la necesidad de agrupar dicha información en un conjunto de datos que permita realizar búsquedas más rápidas, para ello se crearon técnicas que permiten calcular la similitud que tienen dos textos; una de las más utilizadas es la minería de datos, la cual, como su nombre lo indica, se encarga de extraer las partes importantes de un archivo (en nuestro caso un texto). El enfoque que le dimos al proyecto está delimitado precisamente por esta área de la ciencia de datos, la cual nos va a permitir crear varios set de documentos y determinar el número de sets apropiados para agrupar la información, esto gracias a diferentes experimentos, además de generar un informe detallado evaluando el contraste (figura ,**??**) de compartamientos entre el tiempo de ejecución del programa en su implementacion para HPC y el algoritmo implementado con Spark con diferentes datasets.

En este caso, decidimos hacer que la agrupación de documentos sea mediante el uso de los algoritmos k-means y TF-IDF, estos son el principal soporte para determinar la similitud entre documentos.

1. Marco Teórico

K-means Algorithm "The k-means method has been shown to be effective in producing good clustering results for many practical applications. However, a direct algorithm of k-means method requires time proportional to the product of number of patterns and number of clusters per iteration. This is computationally very expensive especially for large datasets. The number of iterations required can vary in a wide range from a few to several thousand depending on the number of patterns, number of clusters, and the input data distribution. Thus, a direct implementation of the k-means method can be computationally very intensive. This is especially true for typical data mining applications with large number of pattern vectors" [1].

The algorithm is composed of the following steps: Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.

Assign each object to the group that has the closest centroid.

When all objects have been assigned, recalculate the positions of the K centroids.

Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

figura sacada del mismo paper. [2] The number of K is determinate by the user and indicate the number of clusters.

Document Clustering Is the act of collecting similar documents in sets. "Document clustering involves the use of descriptors and descriptor extraction. Descriptors are sets of words that describe the contents within the cluster. Document clustering is generally considered to be a centralized process. Examples of document clustering include web document clustering for search users. A web search engine often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by e.g. open source software such as Carrot2. " [3] "Agglomerative hierarchical clustering and K-means are two clustering techniques that are commonly used for document clustering. Agglomerative hierarchical clustering is often portrayed as 'better' than K-means, although slower. " [4]

HPC "High-performance computing (HPC) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to systems that function above a teraflop or 1012 floating-point operations per second. The term HPC is occasionally used as a synonym for supercomputing, although technically a supercomputer is a system that performs at or near the currently highest operational rate for computers. Some supercomputers work at more than a petaflop or 1015 floating-point operations per second." [**?**] This includes "using the world's fastest and largest computers to solve large problems." [5] "High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business." [**?**]

Parallel computing Is a technique that consists of execute many calculations at the same time to solve big problems in less time than it could take in conventional computing. There are different kinds of parallel computing: "Instruction-level parallelism (ILP): Multiple instructions from the same instruction stream can be executed concurrently this kind is limited in practice by data and control dependences. Thread-level or task-level parallelism (TLP): Multiple threads or instruction sequences from the same application can be executed at the same time those could be generated by compiler/user and managed by compiler and hardware and is limited in practice by communication/synchronization overheads and by algorithm characteristics. Data-level parallelism (DLP): Instructions from a single stream operate concurrently on several data is limited by non-regular data manipulation patterns and by memory bandwidth Transaction-level parallelism: Multiple threads/processes from different transactions can be executed concurrently and is limited by concurrency overheads"[6].

Overhead (computing) "The amount of time required to coordinate parallel tasks, as opposed to doing useful work. Parallel overhead can include factors such as: - Task start-up time - Synchronizations - Data communications - Software overhead imposed by parallel languages, libraries, operating system, etc. - Task termination time" [5]

2. Implementación

La implementación de esta practica fue realizada en python ya que es un lenguaje de programación muy sencillo de utilizar pero principalmente por que es un lenguaje sobre el cual se tiene mas experiencia. Para la implementacíon del codigo serial se utilizaron algunas bibliotecas como sys, math, numpy, pyspark.sql y pyspark.mllib que nos permitieron realizar operaciones como lectura de ficheros, manejo de argumentos y brindar caracteristicas para el ordenamiento de alguas estructuras de datos, tambien se empleo numpy como un soporte para las estructuras de datos necesarias para la matriz que contenia todas las distancias entre documentos.

A continuación compartimos nuestra implementación alojada en Github, en la cual basamos el análisis presentado en este estudio; al interior de este link podra encontrar un Readme.md con todas las instrucciones necesarias por poder correr satisfactoriamente los diferentes algoritmos: Repositorio de Github

3. Análisis de resultados (secuencial vs. paralelo)

A continuación presentamos el gráfico resultante del proceso de medición, el cual evidencia las aceleraciones entre datasets e implementaciones y en distintas ejecuciones.

Table 1. Tabla de Comparaciones de tiempo

SetSize (MB)	T. Spark (seg)	T. Paralelo (seg)
119,92	107	307,68
94,88	341,16	216,81
75,02	248,02	174,89
68,66	184,85	142,80

Se puede evidenciar la diferencia entre ejecutar el algoritmo de forma paralela a ejecutarlo de manera serial, aunque el algoritmo paralelo tiene un mayor orden de complejidad, logra reducir el tiempo que tarda en dar la solución, ya que el procesamiento se realiza de una manera distribuída

Conclusiones

-

References

[1] Alsabti, K., Ranka, S. and Singh, V. (2017). An Efficient K-Means Clustering Algorithm. [online] pp.1-2. Available clicking here [Accessed 22 Oct. 2017].

[2] Home.deib.polimi.it. (2017). Clustering - K-means. [online] Available clicking here [Accessed 22 Oct. 2017].

[3] En.wikipedia.org. (2017). Document clustering. [online] Available clicking here [Accessed 22 Oct. 2017].

[4] Glaros.dtc.umn.edu. (2017). A Comparison of Document Clustering Techniques — Karypis Lab. [online] Available clicking here [Accessed 22 Oct. 2017].

[5] Computing.llnl.gov. (2017). Introduction to Parallel Computing. [online] Available clicking here [Accessed 22 Oct. 2017].

[6] Types of Parallelism. [online] Available clicking here [Accessed 22 Oct. 2017].