Clústering de Documentos a partir de Métricas de Similitud

Dillan Alexis Muñetón Avendaño, dmuneto1@eafit.edu.co - Universidad EAFIT
Juan Fernando Ossa Vásquez, jossava@eafit.edu.co - Universidad EAFIT

Este proyecto está basado en minería de texto, donde lo que se quiere es agrupar documentos los cuales deben ser parecidos por su contenido. Se desarrollaron dos programas para esto, uno en programación serial y otro en paralela. Esto con el fin de poder comparar estos tipos de programación y concluir de qué forma y en qué casos es mejor uno que el otro. Se utilizaron algoritmos de similitud y agrupamiento, tales como Jaccard y K-means. El dataset se obtuvo de Gutenberg por medio del siguiente link, https://goo.gl/LL4CgA, en este dataset se encuentran alrededor de 3000 documentos con los cuales se pueden correr en estos programas.

1. PALABRAS CLAVE

(1) HPC: el campo de computación de alto rendimiento (High performance Computing o HPC en inglés) es una herramienta muy importante en el desarrollo de simulaciones computacionales a problemas complejos.
(2) Cluster: conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora
(3) K-Means: es un método de agrupamiento, que tiene como objetivo la partición de un conjunto de n observaciones en k grupos.
(4) Jaccard: algoritmo que mide el grado de similitud entre dos conjuntos.
(5) Matriz: una matriz es un arreglo bidimensional de números. Dado que puede definirse tanto la suma como el producto de matrices, en mayor generalidad se dice que son elementos de un anillo
(6) Python: es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible
(7) mpi4py: es un estándar para python que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.
(8) Programación paralela: Estos generalmente se pueden dividir en clases basadas en las suposiciones que se hacen sobre la arquitectura de memoria subyacente: compartida, distribuida, o compartida-distribuida.
(9) Stopwords: lista de palabras que no deberían ser tomadas en cuenta al momento de comparar los documentos.

2. INTRODUCCIÓN

Este es un proyecto desarrollado por dos estudiantes de la universidad EAFIT, el cual surge como proyecto de una materia la cual pretende exponer cómo la computación de alto rendimiento mejora tiempos de ejecución. Este documento contiene un resumen en cual se da una breve síntesis de lo que se hizo, unos objetivos claros acerca del proyecto. También se muestra el diseño de la solución pensada y planteada por el equipo desarrollador.

El propósito de este texto es mostrar todo lo relacionado con la actividad, teniendo en cuenta cada uno de los elementos los cuales fueron necesarios y utilizados para dicha práctica.

Finalmente, se tuvo en cuenta ciertas recomendaciones hechas por parte del equipo experto o equipo docente que llevó de la mano este proyecto.

3. MARCO TEÓRICO

La minería de texto (text mining), es una de las técnicas de análisis de textos que ha permitido implementar una serie de aplicaciones muy novedosas hoy en día. Buscadores en la web (Google, Facebook, Amazon, Spotify, Netflix, entre otros), sistemas de recomendación, procesamiento natural del lenguaje, son algunas de las aplicaciones.

¿Para qué sirve la Minería de Textos o Text Mining?

Es muy útil para todas la compañías, administraciones y organizaciones en general que por las características propias de su funcionamiento, composición y actividades generan gran cantidad de documentos y que están interesadas en obtener información a partir de todo ese volumen de datos. Les puede servir para conocer mejor a sus clientes, cuáles son sus hábitos, preferencias.

Las técnicas de agrupamiento de documentos (clustering) permiten relacionar un documento con otros parecidos de acuerdo a alguna métrica de similaridad. Esto es muy usado en diferentes aplicaciones como: Clasificación de nuevos documentos entrantes al dataset, búsqueda y recuperación de documentos, ya que cuando se encuentra un documento seleccionado de acuerdo al criterio de búsqueda, el contar con un grupo de documentos relacionados, permite ofrecerle al usuario otros documentos que potencialmente son de interés para él.

Se tomó como base el artículo: Similarity Measures for Text Document Clustering con el objetivo de realizar su paralelización. En este paper se encontraron todas las herramientas y contextualizaciones necesarias para realizar la implementación.

Se tiene un conjunto de documentos grande D, el cual contiene di documentos de texto, se obtuvieron del siguiente link: La base de textos de Gutenberg http://www.gutenberg.org

Las características que debe contener el dataset es que presenten relaciones sintácticas y semántica entre los documentos, por ejemplo en un dataset de noticias presenta una correlación entre documentos de un mismo tipo de noticia (género).

De lo anterior, se derivan dos problemas candidatos a ser paralelizables:

Problema 1: diseño e implementación de una función de similaridad entre dos documentos.

Problema 2: Una vez se tiene definida e implementada la función de similaridad, se puede ejecutar el algoritmo de agrupamiento (clustering), que permite dividir el conjunto de documentos en un número de subgrupos.

Estos problemas se pueden atacar de varias maneras, sin embargo es necesario establecer un punto de partida. El primer acercamiento es analizar la similaridad de cada documento con los demás documentos por medio de un algoritmo serial. Este algoritmo establecerá la línea base para poder comparar las aceleraciones obtenidas a partir de las estrategias de cómputo paralelo en términos de tiempo de ejecución y recursos consumidos. Llegados a este punto, se establece como se deber realizar el particionamiento funcional y/o por datos. Se tuvo en cuenta que no es solo hallar la distancia entre los documentos sino que se debía encontrar los distintos clústers de los

subgrupos de documentos, es decir, sus resultados deberán diferenciar a qué clúster pertenece cada documento a partir de su similaridad. La idea fue diseñar una solución donde se reduzcan los tiempos para el análisis de cada uno de los documentos con respecto a los demás documentos y poder hacer un comparativo teniendo como línea base el tiempo de procesamiento de este problema en un acercamiento secuencial y centralizado con respecto a uno paralelo y distribuido, haciendo un análisis de las herramientas, la estrategia, hallando la aceleración del algoritmo secuencial con respecto al paralelo y con respecto al distribuido.

4. ANÁLISIS Y DISEÑO (PCAM)

En general el esquema de paralelización que se siguió en todo el programa es representado en el siguiente diagrama: Se utilizó esta implementación debido a la facilidad

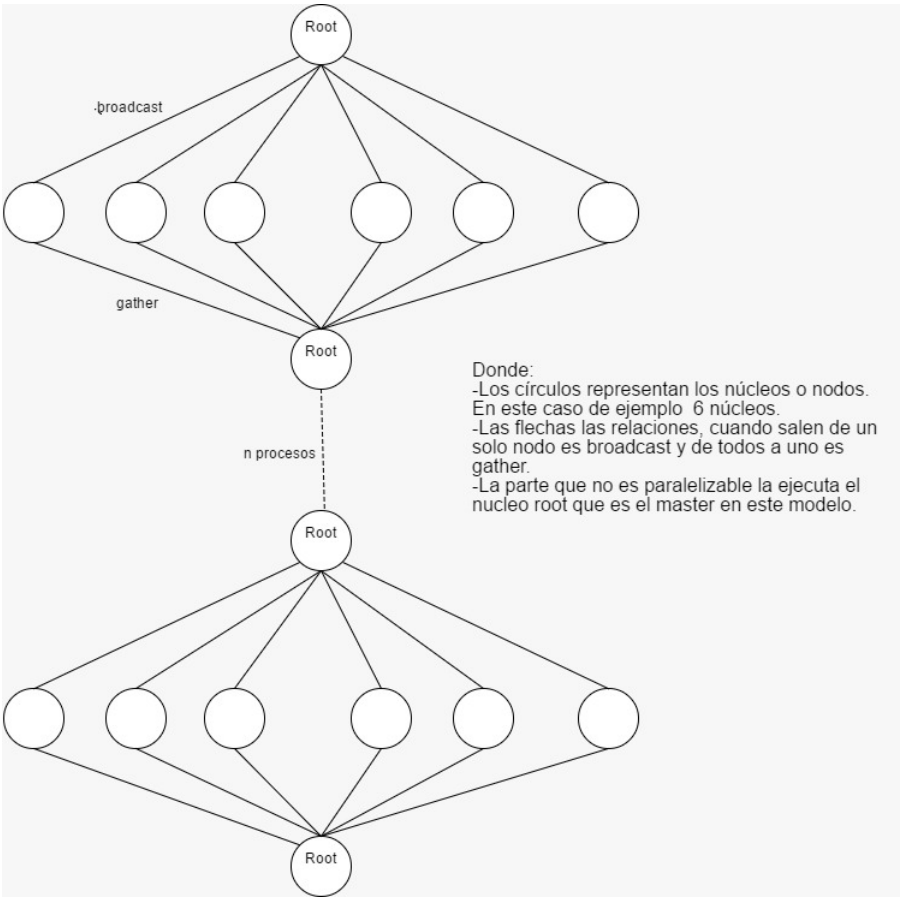


Fig. 1. Diagrama de solución

para ser implementada, pues hay estrategias más eficientes pero con una curva de aprendizaje más alta, donde el tiempo es esencial y para la elaboración de este no se tenía mucho tiempo para aprender las tecnologías así que era necesario utilizar algo eficiente, trabajable y fácil de aprender.

Para la similaridad de los documentos se utilizó Jaccard pues este tenía muy buenas referencias y documentación al respecto, como se muestra en las siguientes tablas:

Table 2: Purity Results

Data	Euclidean	Cosine	Jaccard	Pearson	KLD
20news	0.1	**0.5**	**0.5**	**0.5**	0.38
classic	0.56	0.85	**0.98**	0.85	0.84
hitech	0.29	0.54	0.51	**0.56**	0.53
re0	0.53	**0.78**	0.75	**0.78**	0.77
tr41	0.71	0.71	0.72	**0.78**	0.64
wap	0.32	0.62	**0.63**	0.61	0.61
webkb	0.42	0.68	0.57	0.67	**0.75**

Fig. 2. Purity results

Table 4: Entropy results

Data	Euclidean	Cosine	Jaccard	Pearson	KLD
20news	0.95	**0.49**	**0.51**	**0.49**	0.54
classic	0.78	0.29	**0.06**	0.27	0.3
hitech	0.92	0.64	0.68	0.65	**0.63**
re0	0.6	0.27	0.33	0.26	**0.25**
tr41	0.62	0.33	0.34	**0.3**	0.38
wap	0.75	**0.39**	0.4	**0.39**	0.4
webkb	0.93	0.6	0.74	0.61	**0.51**

Fig. 3. Entropy results

Las explicación del significado de las mismas puede ser encontrada en el artículo “Similarity Measures for Text Document Clustering”. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.4480&rep=rep1&type=pdf

Este algoritmo se obtuvo de la siguiente página: http://dataconomy.com/2015/04/implementing-the-five-most-popular-similarity-measures-in-python/ Aquí se puede entender claramente como funciona el mismo.

Por otro lado, el método de agrupamiento fue K-means, pues fue el elegido por el profesor de la materia quien fue el coordinador del proyecto.
Este algoritmo tiene una implementación estándar, su comportamiento es mostrado en la siguiente gráfica:

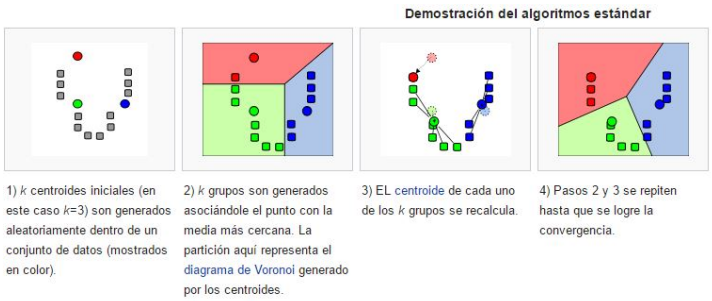


Fig. 4. K-means demostration

Este algoritmo se hizo basado en el algoritmo del siguiente link: https://gist.github.com/bistaumanga/6023692 Aquí se puede entender claramente cómo funciona el mismo.

Las estructuras de datos más utilizadas en el análisis fueron, mapas, colecciones y arreglos, esto es debido a la facilidad con la que python, que fue el lenguaje utilizado, manipula estos tipos de datos. Además este permitía tener un orden de lo que se iba desarrollando, esto será más evidente en el tema de implementación.

5. IMPLEMENTACIÓN

El código de la implementación se puede encontrar en el siguiente respositorio de github: https://github.com/jossava/proyectoTopicosHPC.git

Fig. 5. Implementación

Allí se podrán encontrar 4 carpetas y dos documentos. El documento proyecto3.pdf muestra el problema sobre el cual se trabajo, con una serie de requisitos y restricciones impuestas por el profesor orientador del curso. En la carpeta ejemplosClase están algunos ejemplos de mpi4py utilizados en las clases para la explicación de esta biblioteca. En la carpeta ejemplosOpenMPcpp se situan otros ejemplos, los cuales fueron hechos para probar MPI tanto en Python como en c++, y así tomar la decisión de cuál utilizar. Se podrá encontrar en la carpeta parallelCode y serialCode los archivo .py en los cuales está la implementación del proyecto en programación paralela y serial respectivamente.

Para la ejecución de los mismos se requiere python2.7.x

— Serial

Cuando se requiera ejecutar el programa en serial se debe estar en la carpeta donde se encuentra se debe poner la siguiente línea, $ python2.7 serialCode.py ./{Carpeta donde se encuentran los documentos}/

Para esta ejecución se requiere tener instalado mpi4py

— Paralelo

Cuando se requiera ejecutar el programa en paralelo se debe estar en la carpeta donde se encuentra poner la siguiente línea, $ mpiexec -np {número de nucleos} python2.7 parallelCode.py ./{Carpeta donde se encuentran los documentos}/

Los métodos utilizados en el código fueron los siguientes:

Para el código en paralelo:

```
# Se extraen las palabras mas relevantes
def getMainWords(v):···

# Encuentra las veces que se repite cada una de las palabras de T en cada archivo.
def findT(w):···

# Se llena toda la matrix con las distancias entre pares de documentos.
def jaccardDistances(x):···

# se tomo de http://dataconomy.com/2015/04/implementing-the-five-most-popular-similarity-measures-in-python/
def jaccard_similarity(x, y):···

# Se agrega a C el centroide en el que queda cada documento.
def kMeansC(mJack, cent):···

# Se reubican los centrodes teniendo en cuenta el promedio de sus documentos.
def kMeansRc(z):···
```

Fig. 6. Paralelo

Para el código en serial:

```
# Se extraen las palabras mas relevantes
def getMainWords(rootDir):···

# Encuentra las veces que se repite cada una de las palabras de T en cada archivo.
def findT(T):···

# Se llena toda la matriz con las distancias entre pares de documentos.
def jaccardDistances(findT):···

# se tomo de http://dataconomy.com/2015/04/implementing-the-five-most-popular-similarity-measures-in-python/
def jaccard_similarity(x, y):···

# este metodo se hizo con base en https://gist.github.com/bistaumanga/6023692
def kMeans(X, K, maxIters=10):···


if __name__ == '__main__':···
```

Fig. 7. Serial

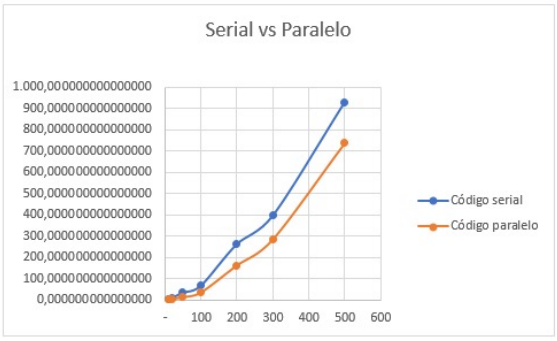
Estos se pueden encontrar en el github. https://github.com/jossava/proyectoTopicosHPC.git

6. ANÁLISIS DE RESULTADOS

Se realizaron pruebas utilizando diferentes mquinas y cantidades de archivos, tanto en serial como en paralelo. Los archivos fueron tomados del dataset de Gutenberg, siempre tomando para las pruebas desde el primero hasta la cantidad requerida.

En la siguiente imagen vemos los resultados de 7 pruebas hechas en un computador HP-240G4 con procesador intel-core i5-5200u de 2 núcleos, con los programas en serial y en paralelo.

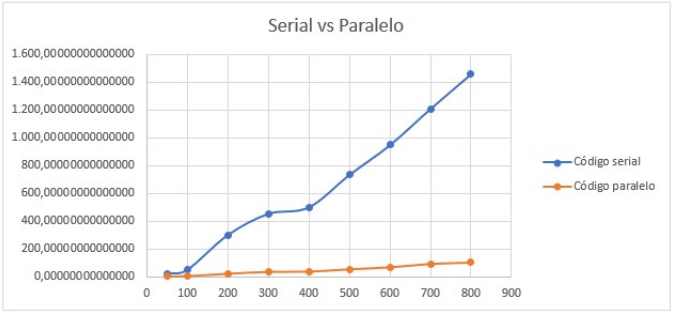
	Código serial	Código paralelo
10	2,773999929428100	1,345000028610220
20	9,378999948501580	3,799999952316280
50	35,027000188827500	13,450999975204400
100	66,870000123977600	34,210000038146900
200	261,903000116348000	161,823999881744000
300	396,088000059127000	284,154999971389000
500	929,306999921798000	738,845000028610000



Como se observa, la ejecución en paralelo pudo mejorar el rendimiento en cada una de las pruebas con respecto al serial, teniendo una aceleración de 1,37 aprox. Aunque hubo una mejora, esta no fue tan notable, debido a que se cambió el número de procesadores de 1 a 2.

También, en la siguiente imagen vemos los resultados de 9 pruebas hechas en el Data Center Academic, el cual contaba con 72 procesadores (para estas pruebas se utilizaron 50 de ellos), igualmente se realizó con los programas en serial y en paralelo.

	Código serial	Código paralelo
50	23,53712606430050	2,63821196556091
100	47,44556689262390	4,33502888679504
200	298,69979405403100	21,19736099243160
300	452,42493200302100	34,98414897918700
400	499,68310689926100	37,03893685340880
500	733,29880785942000	53,94025397300720
600	948,63079690933200	68,31197404861450
700	1.209,78827095031000	92,73696780204770
800	1.460,47877907752000	102,28973817825300



.

Como se evidencia, la ejecución en paralelo mejoró notablemente el rendimiento, mostrando así una aceleración de 13,59 aprox.
Al comparar los resultados mostrados anteriormente, se puede concluir que para la eficiencia de una aplicación en paralelo es tan importante el hardware como el software, claro esta que debe existir un equilibrio entre ellos, ya que ambos tienen límites pero con una correcta combinación podrías generar los mejores resultados posibles.

7. CONCLUSIONES

— Se utilizó la metodología para el diseño de algoritmos paralelos (PCAM)

— Se aplicaron los conocimientos de programación paralela usando estrategias adecuadas para resolver problemas intensivos en recursos computacionales.

— Se analizaron los resultados entre un acercamiento secuencial y paralelo incluyendo las distintas estrategias y tecnologías asociadas a Computación de Alto Rendimiento.

— Se redujeron tiempos de ejecución utilizando estrategias de desarrollo, tecnologías, herramientas y la infraestructura adecuada para ejecutar aplicaciones que requieran tiempos considerables y que de otra manera no sería posible ejecutarlos en un tiempo razonable.

— Se entendieron las limitaciones a nivel de algoritmia, software y hardware asociadas a distintos problemas computacionales que podrían sortearse a partir de herramientas y estrategias como computación paralela y distribuida.

8. REFERENCIAS

Es.wikipedia.org. (2017). ndice Jaccard. [online] Available at: https://es.wikipedia.org/wiki/%C3%8Dndice_Jaccard [Accessed 18 Oct. 2017].

Es.wikipedia.org. (2017). K-means. [online] Available at: https://es.wikipedia.org/wiki/K-means [Accessed 18 Oct. 2017].

Tkatchuk, R., Borne, D., Mouwerik, J. and Tkatchuk, R. (2017). Implementing the Five Most Popular Similarity Measures in Python - Dataconomy. [online] Dataconomy. Available at: http://dataconomy.com/2015/04/implementing-the-five-most-popular-similarity-measures-in-python/ [Accessed 18 Oct. 2017].

Scikit-learn.org. (2017). sklearn.metrics.jaccard_similarity_score scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.jaccard_similarity_score.html [Accessed 18 Oct. 2017].

Anon, (2017). [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.4480&rep=rep1&type=pdf [Accessed 18 Oct. 2017].

Textmining.galeon.com. (2017). Text mining Mineria de Textos Data Mining Mineria de datos recuperacion y organizacion de la informacion. [online] Available at: http://textmining.galeon.com/ [Accessed 21 Oct. 2017].

Gist. (2017). KMeans Clustering Implemented in python with numpy. [online] Available at: https://gist.github.com/bistaumanga/6023692 [Accessed 21 Oct. 2017].