PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
ESCUELA DE POSGRADO

TÍTULO

IDENTIFICACIÓN DE LÍDERES DE OPINIÓN MEDIANTE EL MODELO PROV-DM Y TÉCNICAS DE MINERIA DE GRAFOS

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INFORMÁTICA CON MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

AUTOR

MAURO ANTONIO LEÓN PAYANO

ASESORES

HUGO ALATRISTA SALAS
MIGUEL NUÑEZ DEL PRADO CORTEZ

Junio, 2019

RESUMEN

El análisis de la influencia social nos permite estudiar la manera de determinar la opinión de las personas utilizando como medio el intercambio de información. Dentro de esta disciplina, la identificación líderes de opinión tiene como finalidad identificar a las personas que ejercen un mayor nivel de influencia. La identificación de líderes de opinión se usa en campañas de marketing viral, sistemas de recomendación de productos y en sistemas de detección de anomalías en redes de telefonía móvil. Debido a que los medios sociales se han transformado en la fuente de datos más representativa y relevante para entender el comportamiento de las personas, el análisis de influencia se ha convertido en una de las tecnologías más importantes en las industrias modernas de información y servicios.

Existen diversos métodos para identificar a los líderes de opinión. En este trabajo se plantea un algoritmo híbrido para cuantificar la influencia de acuerdo a atributos estáticos y de interacción de los usuarios pertenecientes a un red social. Los algoritmos híbridos requieren la representación de las interacciones de los usuarios mediante grafos. Por ello, se implementó un algoritmo de construcción, de segmentación y de visualización de grafos con el objeto de abordar los desafíos que involucra identificar y cuantificar la influencia de los usuarios en grandes redes sociales.

El procedimiento fue aplicado en mensajes que tratan sobre el calentamiento global, recolectados desde la plataforma de Twitter con el objetivo de representar en un grafo, a los usuarios interesados en el tema. Los líderes de opinión seleccionados a partir del algoritmo propuesto representan mejor la influencia ganada a través del proceso de difusión.

Este documento consta de 6 Capítulos: El capítulo 1 busca definir el problema y el enfoque adoptado en este trabajo. El Capítulo 2 describe los diversos conceptos, métodos, procesos y herramientas utilizados en el análisis de influencia social tanto en el presente trabajo y estudios relacionados. El Capítulo 3 describe los trabajos previos que busquen identificar líderes de opinión en grandes redes sociales. El Capítulo 4 describe el procedimiento de análisis de influencia social desarrollado. El Capítulo 5 describe los resultados obtenidos en la ejecución del procedimiento propuesto. Finalmente, el Capítulo 6 presentamos las conclusiones y recomendaciones obtenidas producto de trabajo realizado.

"La vida es elegir, puedes elegir ser una víctima o cualquier otra cosa que te propongas."

Dan Millman

Índice general

Lista de Imágenes

Lista de Tablas

Lista de Abreviaturas y Símbolos

OE	**O**bjetivo **E**specífico
ID	**ID**entification
OL	**O**pinion **L**eader
PROV-DM	**PROV**venance **D**ata **M**odel
VADER	**V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner
VAD-SC	**VAD**er **SC**ore
W3C	**W**orld **W**ider **W**eb **C**onsortium
API	**A**pplication **P**rogramming **I**nterface
JSON	**J**ava**S**cript **O**bject **N**otation
SNA	**S**ocial **N**etwork **A**nalysis
OSN	**O**nline **S**ocial **N**etwork
SIA	**S**ocial **I**nfluence **A**nalysis
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
RDD	**R**esilient **D**istribuited **D**atasets
SMS	**S**hort **M**essage **S**ervice
DBSCAN	**D**ensity-**B**asedessage **S**patial **C**lustering of **A**pplications with **N**oise

Dedicado a mis padres y compañeros, por sus consejos y apoyo durante esta aventura.

Capítulo 1

Generalidades

En este capítulo, se describe la necesidad de aplicar un modelo de procedencia de difusión de información en la identificación de lideres de opinión. Además, se presenta la aproximación a dicho problema realizada como parte de la presente tesis. Asimismo, se detallan los objetivos de este trabajo y los resultados que se espera obtener.

1.1. Identificación del problema

El poder de influir en las opiniones e interés de las personas juegan un rol importante en los diferentes entornos de colaboración. Por consiguiente, el estudio de la influencia de usuario es de gran interés de los investigadores en diferentes áreas de como marketing [1], gobierno [2], investigación [3], entre otros. Al inicio, los investigadores utilizaron técnicas cualitativas, como la observación, encuestas y grupos de discusión. Sin embargo, los resultados capturan una visión general de la opinión de las personas. El uso de métodos cuantitativos cobró importancia gracias al trabajo realizado por Lazarsfeld et al.[4]. En su momento, los métodos cuantitativos ayudaron en la definición de métricas para medir la influencia de las personas. No obstante, este método estaba limitado al estudio de la influencia en pequeños grupos de personas involucradas en un evento particular, como las elecciones presidenciales, los canales de ventas, entre otros; debido a los altos costos que implicaba preparar y ejecutar encuestas por evento.

Hoy, el estudio de la influencia de usuarios es una tarea compleja en aspectos de volumen, variedad, velocidad y veracidad [5] de la información generada por los usuarios. Desde la aparición de las plataformas de redes sociales Twitter[1] (fundada el 2003) y Facebook[2] (fundada el 2004), las personas cambiaron la interacción física por la interacción virtual. La interacción virtual, a diferencia de la física, genera grandes volúmenes de datos a través del intercambio de información entre los usuarios; sean fotos, música, vídeos y mensajes u otro tipo de datos, con el fin de expresar su punto de vista acerca de eventos ocurridos en su entorno social. Por lo tanto, las plataformas de redes sociales sean convertido en la principal fuente de información para conocer a los usuarios.

Dada la importancia de estudiar la influencia de los usuarios y la generación continua de grandes volúmenes de datos, los investigadores afrontan nuevos retos en el análisis de influencia social en grandes volúmenes de datos, Social Big Data [6]. A partir de ello, varios investigadores están proponiendo nuevos métodos y algoritmos bajo el enfoque distribuido-paralelo, como los empleados por Zhou et al. [7] y Peng et al. [8]. Sin embargo, existen otros retos más importantes no abordados.

Los usuarios reaccionan de diferentes formas por cada mensaje, por lo tanto, en una red social es importante conocer el sentido (positivo o negativo) en que son influenciados los usuarios por los líderes de opinión. Además, las redes sociales reales son complejas y están conformadas por múltiples tipos de entidades, enlaces, atributos estáticos y de interacción [9]. Por lo tanto, es necesario contar con un proceso para analizar la influencia de usuarios en redes sociales heterogénea. Por último pero no menos importante, la

[1]Twitter: https://twitter.com/
[2]Facebook: https://www.facebook.com/

procedencia de datos es un concepto clave que permite identificar el origen de los datos y sus movimientos durante el tiempo [10]. En el contexto de análisis de influencia social, no necesariamente los usuarios influyentes son autores de la información que difunden, por lo tanto, identificar a los usuarios que crean o comparten información influyente o viral es un factor clave en la generación de influencia en las redes sociales de usuarios.

Por lo mencionado, en el presente trabajo se plantea un proceso de análisis de influencia social, incluyendo la procedencia de la información influyente y el sentido de la influencia (positivo y negativo) aplicable a una red social heterogénea. A través del proceso propuesto, identificaremos a los líderes de opinión (usuarios más influyentes) utilizando un algoritmo híbrido implementado en Apache GraphX[3].

1.2. Objetivo general

El objetivo general del presente trabajo es implementar un proceso que permita la identificación de usuarios influyentes en una red social utilizando un algoritmo híbrido en un entorno de procesamiento en paralelo (Big Data).

1.3. Objetivos específicos

Los objetivos específicos para este trabajo son:

OE1: Implementar un componente de software para la extracción y almacenamiento de comentarios de redes sociales basados en microblogs.

OE2: Representar los usuarios de una red social mediante una estructura de datos que sea soportado por un lenguaje de programación de alto nivel.

OE3: Seleccionar las comunidades de usuarios significativas.

OE4: Seleccionar los líderes de opinión a partir de características estáticas y las interacciones entre los usuarios.

OE5: Visualizar la influencia de los líderes de opinión en la estructura de datos seleccionada.

[3]Apache GraphX: https://spark.apache.org/graphx/

1.4. Resultados esperados

a) Para el OE1, obtener un conjunto de tweets cuyo contenido está relacionado al calentamiento global.

b) Para el OE2, un algoritmo para la construcción de una red social, que permita construir un grafo dirigido a partir del conjunto de datos del resultado esperado 1.

c) Para el OE3, un algoritmo de detección de comunidades de usuarios, que permita segmentar la red social e identificar comunidades significativas.

d) Para el OE4, un algoritmo híbrido, que permita identificar a los líderes de opinión en cada comunidad significativa.

e) Para el OE5, un componente software de visualización de grafos, que permita analizar la influencia ejercida por los líderes de opinión.

1.5. Justificación

De acuerdo a estudio realizados por Hootsuite [11] en el 2018, el 42 % de la población mundial utilizan activamente los medios sociales para compartir información dentro de sus redes sociales. Debido a ello, las plataformas de medios sociales se han convertido en la fuente de datos más relevante para conocer e interactuar con los usuarios que se encuentran conectado a internet.

Este punto de vista es compartido por otros investigadores, quienes utilizan las redes sociales existentes en las plataformas de medios sociales para interactuar y generar colaboración entre los usuarios ó como fuente de información para afrontar eventos particulares. Por ejemplo, Avvenuti et al. [12] utilizan la plataforma de Twitter como fuente de información para identificar los daños provocados por el terremoto ocurrido en Italia en la ciudad de Finale Emilia en el 2015. Por otro lado, Gao et al. [13] plantean el uso de las plataformas de medios sociales para apoyar las tareas de asistencia, coordinación y difusión de información, con el fin afrontar las situaciones de emergencias ocasionadas por los desastres naturales. Ambos trabajos, manifiesta la importancia de contar con la información oportuna para reaccionar correctamente ante una situación cercana y en identificar agentes de cambio que faciliten la difusión de información a la población vulnerable.

Por lo tanto, en el presente trabajo se enfoca en afrontar los retos relacionados al análisis de influencia social en grandes volúmenes de datos, considerando la procedencia de la información, la heterogeneidad de entidades en las redes sociales y el sentido de la

influencia propagada. Además, proponemos un algoritmo híbrido (atributos estáticos y de interacción) para cuantificar la influencia de los usuarios con el fin de identificar líderes de opinión que apoyen en la difusión oportuna de información.

Debido a la diversidad de eventos que ocurren en la sociedad, la participación de los usuarios e información generada en los medios sociales, en el presente trabajo se toman ciertas consideraciones explicadas en la siguiente sección.

1.6. Límites del proyecto

Debido a la amplia gama de temas o tópicos discutidos por los usuarios de redes sociales, el presente trabajo esta limitado a estudiar a los usuarios interesados en el calentamiento global. Este tema fue elegido porque es un problema mundial, y por el afán personal de contribuir en la identificación de los líderes de opinión (los usuarios más influyentes) que puedan impulsar la generación de consciencia sobre el tema.

En el trabajo estudiamos la red social generada por los usuarios de Twitter que comentan sobre el calentamiento global. Por lo tanto, la población de estudio esta limitado a los datos recolectados por el API de Streaming de Twitter [14], del 03/07/2018 al 23/09/2018. De acuerdo con Piper [15], los tweets recolectados mediante el API de Streaming representan el 100 % de la población total de tweets generados en la plataforma de Twitter entre las fechas indicadas, en constraste al análisis realizado en la sección 5.1.

Por último, los recursos utilizados en el proceso de experimentación esta limitado al uso de una computadora portátil de trabajo de 16 GB de memoria RAM y 8 unidades de CPU. Además, debido al uso simultáneo de Apache Spark, como motor de procesamiento distribuido en memoria, y Apache Hadoop, como almacenamiento distribuido de datos, los recursos disponibles para el procesamiento de nuestros datos se reducen a 8 GB de memoria RAM y 6 unidades de CPU.

1.7. Aportes del proyecto

El presente trabajo hace énfasis en la importancia del estudio de la influencia social en grandes redes sociales. Por consiguiente, nuestro principal aporte es la definición de un proceso de análisis de influencia social en grandes redes sociales heterogéneas, definido en la sección 2.3, a través del cual aportamos artefactos y definiciones reutilizables para futuros trabajos. Primero, se propone un componente software para extraer continuamente datos desde la plataforma de Twitter, a través de la selección de tópicos. Segundo,

se propone el uso del modelo de procedencia de datos PROV-DM [16] para representar la red social a través de un grafo dirigido con nodos y enlaces heterogéneos, con el fin de caracterizar la difusión de información e analizar la influencia indirecta. Tercero, se propone un algoritmo híbrido distribuido para cuantificar la influencia directa e indirecta que ejercen los usuarios dentro de las comunidades existentes en el grafo heterogéneo previamente construido. A pesar de contar con límites para el procesamiento distribuido de datos, el algoritmo propuesto está definido para que pueda procesar datos en paralelo y distribuidos en una o más máquinas con Apache Spark[4]. Por último, se propone un componente software para visualizar los resultados de la construcción de la red social y la identificación de comunidades significativas; y estudiar el comportamiento de los líderes de opinión identificados por el algoritmo propuesto.

[4]Apache Spark: https://spark.apache.org/

Capítulo 2

Marco conceptual

En este capítulo, exploramos los conceptos entorno a la identificación de usuarios influyentes en redes sociales describiendo algunas de las principales técnicas existentes sobre medición de influencia. Asimismo, profundizamos en los conceptos de los modelos, herramientas de procesamiento distribuido y técnicas complementarias, utilizadas en el proceso de análisis de influencia social propuesto.

2.1. Líder de opinión

Es la persona, grupo o entidad que tiene la capacidad de influir en la opinión otras personas, debido a ciertas cualidades tales como: experiencia, habilidad y/o reputación. No se consideran a priori, líderes de opinión; a las personas, grupos o entidades que ejercen un cargo de líderes, tales como jefes de empresa o jefes de Estado [17].

2.2. Análisis de influencia social

El estudio de la influencia social (SNA, por sus siglas en inglés) se ha convertido en la investigación más importante sobre las redes sociales, como Facebook, Twitter, LinkedIn[1] y Google Plus[2], debido a que se convirtieron en las fuentes de datos más representativas y relevantes para estudiar el comportamiento de las personas [6].

El análisis de influencia social esta conformado por procesos y métodos, de minería de datos, aprendizaje de máquina, estadísticos, minería de grafos, lingüística, procesamiento

[1]LinkedIn: https://www.linkedin.com/
[2]Google Plus: https://plus.google.com/discover

de lenguaje natural, web semántica, ontologías y procesamiento de grandes volúmenes de datos (Big Data), en busca de cuantificar la influencia de cada usuario y como identificar a los usuarios más influyentes en una red social.

Muchos de los estudios sobre la influencia social se realizaron con datos de las plataformas de redes sociales como Facebook, Twitter y Sina Weibo[3], e incluyen tareas de análisis y procesamiento de datos, evaluación de influencia, identificación de usuarios influyentes, modelado de la difusión de información y principalmente, en la cuantificación de la influencia de los usuarios [9].

Actualmente, existen diversas formas para calcular la influencia que ejerce un usuario de una red social [18]. Esta diversidad de propuestas se debe a dos factores: un problema conceptual y a la naturaleza de los datos utilizados en el análisis. Desde el punto vista conceptual, de acuerdo de la definición de "usuario influyente", los usuarios se categorizan en: líder de opinión, influenciador y comentarista por el tipo de actividad e impacto producido [19]; usuario emisor o pasivo de acuerdo a su popularidad [20]; y usuario principal, usuario regular, usuario zombie o usuario aislado por la transferencia de influencia [21].

Por otro lado, otras propuestas depende de los datos utilizados en el análisis. Zhu et al. [22] utilizan datos recolectados de Pinterest[4] para calcular la influencia de los usuarios a través del atributo de publicación de contenido (*pin*) y el atributo de compartición de un pin que otro usuario a publicado (*repin*). Cheng et al. [23] utilizan datos recolectados de la pagina de noticias deportivas Mobile01[5] para calcular la influencia utilizando como atributos: el número total de artículos publicados por un usuarios (*article_num*), la probabilidad de que un artículo sea correspondido por otro usuario (*replied_by_prob*), el grado de experiencia de un usuario publicando contenido de un dominio específico (*expert_deg*) y la probabilidad de que los artículos de un usuario respondan a otros usuarios (*reply_prob*). Por otro lado, Peng et al. [8] usan datos de telecomunicaciones (SMS/MMS) para crear un grafo bidireccional que permita obtener los atributos de interacción necesarios en el cálculo de la influencia.

Los trabajos anteriores evidencian que el estudio de la influencia social está fuertemente relacionada a las características de la red social estudiada. Por ende, los atributos utilizados para calcular la influencia en una red social X no pueden utilizarse para calcular la influencia en una red social Y, debido a la ausencia de datos (atributos) y a las diversas formas de analizar una red social (a través de atributos y/o estructuras). Por lo comentado, identificamos tres formas de cuantificar la influencia de los usuarios a través

[3]Sina Weibo: https://www.weibo.com/
[4]Pinterest: https://www.pinterest.es/
[5]Mobile01: http://www.mobile01.com

de sus atributos estáticos, atributos de interacción y ambos tipos de atributos (enfoque híbrido).

A continuación se describen los algoritmos de cálculo de influencia social, clasificados de acuerdo al tipo de atributos que utilizan en sus cálculos.

2.2.1. Influencia en base a atributos estáticos

Este conjunto de algoritmos utilizan los atributos estáticos del usuario o de los objetos generados en la red social (mensajes, texto, vídeos o imágenes) extraído de las fuentes de datos. Los atributos estáticos reflejan un valor en un instante de tiempo específico, atributos como: nombre, fecha de creación e identificador único de usuario, son ejemplos típicos de esta clase de atributos. Principalmente, son utilizados por investigadores que utilizan técnicas de aprendizaje de máquina, estadísticas, minería de datos y procesamiento de lenguaje natural en el estudio de redes sociales.

Por ejemplo, Ding et al. [24] usan los atributos: número de usuarios que siguen cada usuario y número de usuarios seguidos por cada usuario, obtenidos de la red social Sina Weibo[6], para identificar a los 100 usuarios más influyentes aplicando el algoritmo de agrupación K-means. En total se procesaron datos de 5 025 cuentas de usuarios pertenecientes a estudiantes de la Universidad de Shanghai. Erlandsson et al. [25] aplican el método de reglas de asociación para identificar a los usuarios más influyentes. Utiliza los datos recolectados desde páginas públicas de Facebook y los filtra por los atributos como: estado del usuario, número de usuarios, número de post y el número de comentarios. Lin et al. [26] utilizan como atributos: el número comentarios enviados y el número de respuestas a los comentarios enviados de cada usuario identificado entre los 4 406 registros extraídos desde Sina Weibo. Para el cálculo de la influencia, proponen el algoritmo DBSCAN [27], el cual aplica una técnica de agrupación sin necesidad de indicar el número de grupos para identificar a los usuarios más influyentes.

2.2.2. Influencia en base a atributos de interacción

Los atributos de interacción, son aquellos atributos que representan la interacción que realiza un determinado usuario con otros usuarios u objetos en una red social. Para ello, los algoritmos utilizan un grafo dirigido o no dirigido para representar la red social de usuarios a estudiar. El uso de los grafos permiten generar los atributos de interacción presente entre los usuarios. Por ejemplo: el grado de centralidad de un nodo usuario u,

[6]Sina Weibo: http://e.weibo.com/

perteneciente a un grafo homogéneo (grafos con nodos de un solo tipo), representa el número de usuarios que interactúan directamente con el usuario u.

Khan et al. [28] utilizan datos de los tweets extraídos desde Twitter sobre la protesta de un político pakistaní Imran Khan, para crear un grafo homogéneo donde cada nodo representa a un usuario. A partir del grafo de usuarios, define la influencia de un usuario v como la suma de tres métricas de centralidad: el grado de centralidad, la centralidad de intermediación y la centralidad de cercanía del nodo usuario. De acuerdo a valor obtenido de la suma de las tres métricas mencionadas, a cada usuario se le asigna el valor de su influencia y a partir de ella se identifican a los líderes de opinión en el grafo de usuarios.

A partir de los dos tipos de atributos explicados anteriormente, se realizaron trabajos combinando los atributos estáticos y de interacción en la definición de la influencia de los usuarios. Este nuevo enfoque se explicará en el siguiente apartado.

2.2.3. Influencia en base de atributos estáticos y de interacción

Este conjunto de algoritmos utilizan un enfoque híbrido porque utiliza los atributos estáticos y atributos de interacción de los usuarios para calcular la influencia.

En este grupo encontramos el trabajo realizado por Wei et al. [29], quienes usando 96 281 registros de usuarios de la red social Sina Weibo, calculan los atributos de interacción: número de seguidores, número de comentarios enviados entre usuarios, y el atributo estático: contenido del comentario en cada registro de usuario que sirven de entrada en su algoritmo propuesto para calcular la influencia recibida por cada usuario. Chen et al. [30], utilizando datos públicos del servicio de microblog de Tencent Weibo[7], plantean la función de influencia global en una red de usuarios, como se aprecia en la Ecuación 2.1:

$$Influence(v) = (1-\lambda)\frac{Followers(v)}{N} + \lambda \times \sum_{v' \in Followers(v)} \frac{RI(v,v') \times Influence(v')}{Followers(v')} \tag{2.1}$$

Donde la influencia global de un usuario v se calcula como la suma entre dos componentes: la proporción de sus seguidores (Followers(v)) del total de usuarios en la red (N) y el resultado obtenido de sumar por cada usuario v' seguidor del usuario v: la influencia relativa entre los usuarios v y v' (RI(v,v')) multiplicado por la influencia global que ejerce el usuario v' (Influence(v')) entre sus seguidores (Followers(v')). Además, se define una variable lambda (λ) para indicar la relevancia de cada componente.

[7] Tencent Weibo: http://t.qq.com/

Por otra parte, Hajian et al. [31] utilizan una grafo dirigido para representar a los usuarios como nodos y las relaciones entre los usuarios como enlaces. A partir del grafo de usuarios, definen la influencia de un usuario v como se aprecia en la Ecuación 2.2.

$$IR(v) = (1 - \delta(v))\frac{\sum_{v' \in F(v)} IR(v')}{|F(v)|} + \delta(v) \times MOI(v) \tag{2.2}$$

Donde la influencia de un usuario v (IR(v)) se define como la suma de dos componentes: la relación entre la suma de la influencia de todos los seguidores del usuario v y el total sus seguidores, y la magnitud de la influencia ejercida por el usuario v (MOI, por sus siglas en inglés), obtenida a partir de la relación de afecto con sus seguidores (ROA, por sus siglas en inglés). La relevancia de cada componente se indica a través de la popularidad del usuario v en la grafo ($\delta(v)$). Los líderes de opinión son seleccionados del grupo de usuarios con valores IR más altos.

En cada uno de los trabajos revisados, se emplea un conjunto de tareas para transformar los datos obtenidos desde una plataforma de red social en una lista de usuarios influyentes.

2.3. Proceso de análisis de influencia social

El proceso de analizar la influencia social en un red social requiere de un conjunto ordenado de pasos para la obtención de conocimiento. Con este fin, varios autores definen sus propios procesos [23, 32–34], no obstante, consideramos la arquitectura de análisis de influencia social propuesto por Peng et al. [9], junto con el flujo de análisis de datos con Big Data propuesto por Labrinidis y Jagadish [35], en la definición de nuestro proceso de análisis de influencia social en grandes redes sociales como se puede apreciarse en la Figura 2.1.

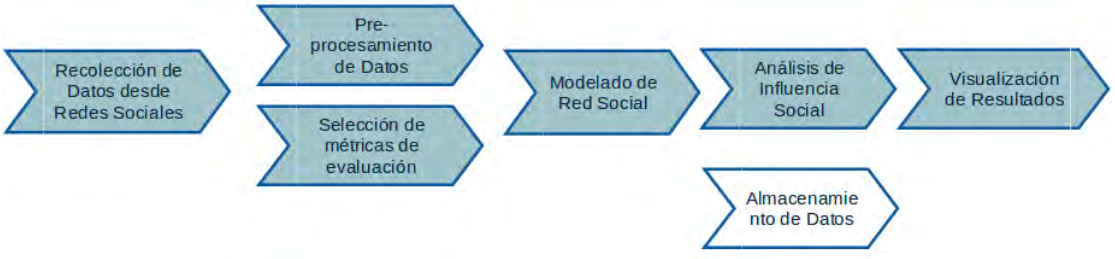


FIGURA 2.1: Proceso de análisis de influencia social.
Fuente: Basado en Labrinidis y Jagadish [5].

Por lo tanto, en el presente trabajo se realiza el análisis de influencia social aplicando las seis etapas operacionales y una etapa de soporte explicadas en las siguientes subsecciones.

2.3.1. Recolección de datos desde redes sociales

Es la etapa más importante en el análisis de la influencia social e involucra la conexión a las plataformas de redes sociales a través de un API para extraer los mensajes compartidos entre los usuarios. En nuestro trabajo elegimos Twitter como fuente de datos para extraer los mensajes creados y compartidos entre sus usuarios. La extracción de los mensajes (de ahora en adelante denominado *tweet*), se realizará a través del API de Streaming de Twitter [14]. A diferencia de las otras APIs de Twitter, el API de Streaming genera una comunicación persistente con un programa cliente permitiéndole extraer los tweets creados y compartidos por los usuarios que reaccionan a los eventos ocurridos en sus entornos.

	user	tweet
user	follows / is followed by mention replies to retweets to	posts retweets likes replies
tweet	posted by retweeted by liked by replied by	replies / is replied from retweets / is retweeted from

Tabla 2.1: Relaciones de Twitter entre usuarios y tweets.
Fuente: Fabián Riquelme and Pablo González-Cantergiani [18].

En Twitter existen cuatro tipos de relaciones: usuario-usuario, usuario-tweet, tweet-tweet y tweet-usuario. En cada tipo de relación participan dos entidades: usuario y tweet, a través de una o muchas acciones definidas entre ellas, como se muestra en la Tabla 2.1. Por ejemplo, un usuario puede mencionar a otros usuarios pero no puede mencionar un tweet ó un tweet puede ser posteado por un usuario pero no postear otro tweet. Estas acciones nos permitirán entender el contexto en que se establecen las relaciones entre los objetos usuarios y tweets en la red social estudiada. Los objetos usuario y tweet tiene un lista pre-establecida de atributos definidos, ver detalle en el Apéndice A. Finalmente, los tweets recolectados se guardan a través nuestro procedimiento de almacenamiento de datos.

2.3.2. Pre-procesamiento de datos

En esta fase se remueve la información irrelevante y redundante con el fin de proporcionar una mejor representación de la información que reduzca el costo computacional en la aplicación de los algoritmos de aprendizaje de maquina.

2.3.2.1. Selección de atributos

Esta técnica busca reducir el número de atributos sin modificaciones a través de la identificación y eliminación de atributos innecesarios o redundantes para obtener atributos relevantes. Aplicar esta técnica tiene beneficios directos como la reducción de los tiempos de lectura de datos y la reducción de complejidad en la construcción del modelo.

2.3.2.2. Selección de instancias

Existen casos en donde no se puede obtener datos consistentes en los atributos relevantes debido a la presencia de valores perdidos, especiales, anómalos e inconsistentes. Para este tipo de situaciones, se empleará la técnica de selección de instancias. La cuál consiste en eliminar los registros inconsistentes a través de expresiones lambda [36].

La descripción de los atributos obtenidos de los tweets, la eliminación de atributos y el filtrado de tweets incompletos se detalla en la Sección 4.2.

2.3.2.3. Sanitización de atributos

La sanitización es el proceso utilizado para manejar los datos confidenciales o sensibles contenido en los tweets obtenidos desde el API de Streaming de Twitter. Los atributos de los tweets se clasificaran como sensibles, si permite identificar al usuario autor del tweet con un cierto nivel de esfuerzo.

Como parte de los atributos sensibles, estudiados en la Sección 4.2, existe un identificador único de los tweets y usuarios creados en Twitter. Estos atributos requieren un tratamiento especial debido a su importancia en el proceso de construcción de la red social y para mostrar los resultados. Por lo tanto, los atributos identificadores pasarán por una función hash no criptográfica para ocultar su valor real y proporcionar un identificador único diferente al valor original.

En esta etapa utilizamos el algoritmo de Murmurhash3 creada por Appleby [37], el cuál genera un valor hash de 64 bits. Murmurhash3 no pertenece a la clase de algoritmos

de criptografía fuerte; por el contrario, se enfoca en proporcionar una valor hash bien distribuido y rápido. Con 64 bits, Murmurhash3 puede generar $2^{N/2}$ valores hash de N bits, por lo tanto, podemos generar 4 294 967 296 valores hash antes de generar una colisión.

Murmurhash3 se encuentra implementado en diferentes lenguajes de programación como Go[8], Python[9], Java[10] y Scala[11]; siendo esta última la implementación utilizada en el presente trabajo por ser el mismo lenguaje de programación utilizado en GraphX de Apache Spark.

2.3.3. Selección de la métrica de evaluación

Esta etapa se centra en la extracción de un conjunto de métricas para caracterizar con precisión el comportamiento de los usuarios en una red social, ya que estas medidas de evaluación son útiles para cuantificar la influencia social de cada usuario.

Las métricas de evaluación utilizadas en el presente trabajo y en otros trabajos mencionados son: la centralidad de grado, la centralidad de Bonacich y la centralidad de intermediación, además se define el concepto de componentes conectados. A continuación detallamos cada uno de las métricas mencionadas.

2.3.3.1. Centralidad de grado

Esta métrica define la centralidad como el número enlaces que posee un usuario con los demás usuarios. Formalmente, dado un grafo $G = (N, E)$, donde N es su conjunto de nodos y E su conjunto de enlaces, la centralidad de grado de un nodo i $\in N$, se define en la Ecuación 2.3.

$$C_{GRADO}(i) = \sum_{i=1}^{n} \alpha_{ji} \tag{2.3}$$

Donde n es el número de nodos j distintos al nodo $i \in N$, α_{ij} asume el valor de 1, si existe el enlace $(i, j) \in E$ y el valor 0, si no existe. En el caso de grafos dirigidos, se pueden definir dos medidas de centralidad de grados diferentes, correspondientes a los enlaces de entrada y de salida de cada nodo.

[8] Go: https://godoc.org/github.com/spaolacci/murmur3
[9] Python: https://pypi.org/project/mmh3/
[10] Java: https://google.github.io/guava/releases/snapshot/api/docs/com/google/common/hash/Hashing.html
[11] Scala: https://www.scala-lang.org/api/current/scala/util/hashing/MurmurHash3$.html

2.3.3.2. Centralidad de Bonacich

Esta métrica define la centralidad como la suma de todas las conexiones entre un nodo i con todos los nodos j que se encuentran conectados con él a través de un camino, al mismo tiempo que se penaliza los caminos con factor de atenuación $\beta \in (-1, 1)$. De acuerdo a la propuesta de Bonacich [38], la centralidad de un nodo i $\in N$ se obtiene aplicando la Ecuación 2.4.

$$C_{BON}(i) = \sum_{i=1}^{n} (\alpha + C_{BON}(j))\alpha_{ji} \tag{2.4}$$

Donde n es el número de nodos j distintos al nodo $i \in N$, α es una constante de normalización y β determina qué nodos j del grafo influyen en el cálculo de la centralidad del nodo i. Si β es pequeño (atenuación alta), los nodos j más cercanos al nodo i influyen en el valor de centralidad, si β es grande (atenuación baja), los nodos de los caminos encontrados influyen en el valor de la centralidad y si $\beta = 0$, la centralidad de Bonacich coincide con la Centralidad de grado.

2.3.3.3. Centralidad de intermediación

Esta métrica cuantifica la frecuencia o el número de veces que un nodo actúa como un puente a lo largo del camino más corto entre otros dos nodos en un grafo. Propuesto por Freeman [39], la centralidad de intermediación (Betweenness Centrality, traducción del inglés) de un nodo $i \in N$ es establecido empleando la Ecuación 2.5.

$$C_{BET}(i) = \sum_{j,k} \frac{b_{jik}}{b_{jk}} \tag{2.5}$$

Donde b_{jk} es el número de caminos más cortos desde el nodo j hasta el nodo k, y b_{jik} el número de caminos más cortos desde j hasta k que pasan a través del nodo i.

2.3.3.4. Componentes Conectados

Son los subgrafos conectados de un grafo no convexo. Cada subgrafo esta conformado por un conjunto de nodos conectados a través de un o varios enlaces $\in E$.

En la Figura 2.2 se tiene 3 componentes conectados. El primero está formado por el nodo A, el segundo está formado por los nodos B, C y D, y el tercer componente está formado por los nodos E y F.

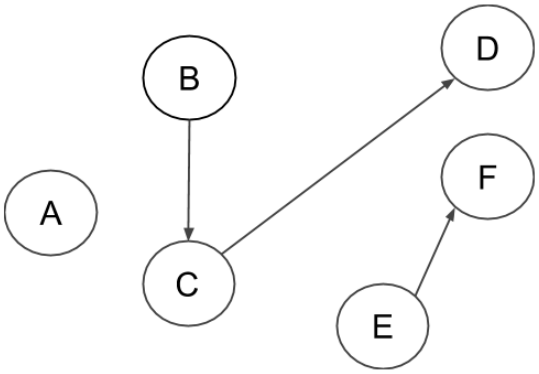


FIGURA 2.2: Grafo no convexo.
Fuente: Elaboración propia.

Tanto las métricas de centralidad como la aplicación del algoritmo para obtener componentes conectados requiere de un grafo como estructura representativa de una red social. El modelado de una red social como un grafo a partir de los tweets se describirá en la siguiente subsección.

2.3.4. Modelado de red social

Esta etapa se centra en la construcción de la red social a través del mapeado de los datos. En este apartado se revisará los métodos de *Red completa* y *Bola de nieve* mencionados en [40].

2.3.4.1. Método de bola de nieve

Este método reconstruye una red a partir de un nodo o conjunto de nodos centrales. Por ejemplo, si se tiene un conjunto de nodos de tipo usuario, se selecciona un nodo usuario del conjunto, denominada nodo principal, y se procede a mapear todos los nodos de tipo usuario y tweet interconectada al nodo principal. Luego, los nuevos nodos de tipo usuario identificados se agregan al conjunto de nodos inicial y se selecciona el siguiente nodo principal para repetir el proceso. El proceso continua hasta que no se pueda identificar nuevos nodos o hasta que se cumpla un criterio de paro.

Este método es útil en el rastreo de grupos de nodos especiales y para estudiar las interacciones de los nodos fuertemente conectados. Sin embargo, tiene la limitante de no poder mapear los nodos o grupos de nodos no conectados (es decir, aislados), siendo una característica importante en el análisis de redes sociales.

2.3.4.2. Método de red completa

Este método permite construir una red social con toda la información recolectada, incluido los nodos o grupos de nodos aislados. Se requiere un conjunto de nodos, identificados dentro de los tweets recolectados. Cada nodo identificado se busca y mapea todos los enlaces existentes con los demás nodos identificados.

Si bien este método recopila todos los enlaces existe entre los nodos, es costoso y difícil de ejecutar [40]. Sin embargo, se vuelve manejable si dividimos la búsqueda en bloques.

En el proceso de construcción se puede mapear dos tipos de redes sociales: la red social de seguidores y la red social de actividades. La red de seguidores se genera al mapear los enlaces permanentes entre los nodos inactivos. El crecimiento de este tipo de red es lento. Contrariamente, la red de actividades se centra en la interacción activa de los nodos y presenta un crecimiento rápido, particularmente cuando se comparte contenido viral.

2.3.5. Análisis de influencia social

Para calcular la influencia de los usuarios de nuestro grafo heterogéneo dirigido, definimos un algoritmo híbrido en función de los atributos estático del objeto *Tweet* y a los atributos de interacción del objeto *User*, seleccionados.

El algoritmo tiene como objetivo cuantificar la influencia que trasmite un usuario a otros usuarios a través de sus tweets. En este contexto, un usuario reacciona a la opinión de otro usuario a través de la generación de otro tweet. En el presente trabajo, consideramos que un tweet puede generarse en dos situaciones:

- Si concuerda completamente con el contenido del tweet, el usuario retuitea el tweet.
- Si concuerda parcialmente con el contenido del tweet y agrega contenido complementaria al tweet original, el usuario cita al tweet.

En caso el contenido de un tweet sea del agrado de otro usuario, el otro usuario reaccionará con un "me gusta" en el tweet, pero esta reacción no genera un nuevo tweet. Por lo tanto, las reacciones de *like* está fuera del alcance del presente trabajo.

En la definición del algoritmo no se consideran los atributos estáticos: número de seguidores del usuario (*followers_count*) y número de amigos del usuario (*friends_count*), debido a que son atributos cuyos valores están relacionados a un contexto más general. En otras palabras, si un usuario tiene 125 seguidores, no necesariamente son seguidores

interesados en el tema del calentamiento global, sino pueden ser seguidores ganados al publicar tweets respecto a un tema ajeno al estudiado. Si bien, el beneficio de tener n seguidores en Twitter permite propagar n veces cada vez que se crea un nuevo tweet, no asegura que todos los seguidores reaccionen igual al tweet.

Los criterios y conceptos descritos en esta sección son la base considerada para la definición del algoritmo híbrido planteado en la Sección 4.5.

2.3.6. Visualización de resultados

Esta etapa se enfoca en mostrar de forma gráfica los resultados obtenidos en la etapa anterior. No obstante, la visualización de grafos de gran tamaño es una tarea desafiante desde el punto de vista computacional, existiendo dos tecnologías: la visualización con motores de renderizado especializado, como Unreal Engine [12] y Unity [13], o la visualización con librerías de renderizado web. Este último, lo conforman librerías JavaScript que utilizan navegadores web, como Opera, Mozilla Firefox y Google Chrome, para visualizar datos. Entre las principales librerías y entornos de visualización web tenemos:

a) Neo4j Browser: utiliza el lenguaje Cypher para generar grafos de manera secuencial mediante la librería de javascripts D3.js [41]. Dado que Neo4j Browser es un componente integrado a la base de datos NoSQL Neo4j, se requiere migrar los datos a su entorno para visualizar los resultados.

b) GraphLab: es un framework enfocado en la ejecución de algoritmos de aprendizaje de máquina en entornos distribuidos [42]. Implementado en Python y optimiza sus operaciones internas a través de C++, proporciona un modelo de análisis de grafos y algoritmos pre-definidos para su ejecución. No obstante, su uso requiere de una licencia académica el cuál solo permite la visualización de mil vértices.

c) D3.js: librería de javascript para la manipulación de documentos web basados en datos. Tiene múltiples ejemplos y buena documentación. Sin embargo, al utilizar la interfaz DOM para el manejo de objetos (nodos y vértices) ralentiza la visualización de grafos grandes [43].

d) Linkurious.js: librería javascript diseñada para la exploración interactiva de grafos grandes en la web. Se basa de la librería Sigma.js y agrega características adicionales. Además, permite la visualización de más de 100 millones de vértices, compatible con los exploradores web modernos y se puede integrar con base de

[12] Unreal Engine: https://www.unrealengine.com
[13] Unity: https://unity.com/

datos orientadas a grafos como Neo4j, JanusGraph, AllegroGraph y Startdog [44]. No obstante, su uso requiere de una licencia comercial.

e) Vivagraph.js: librería de javascript de código libre diseñada para la visualización de grafos utilizando los formatos WebGL, SVG o CSS. Proporciona una exploración interactiva a través del rastreo y renderizado de los cambios realizados en el grafo. Su diseño permite la extensión de sus funcionalidades y admite diferentes motores de renderizado gráficos y algoritmos de otras librerías [45].

En las etapas de pre-procesamiento de datos, modelado de red social y análisis de influencia social se generan datos intermedios que deben persistir durante nuestro proceso de análisis. Debido a ello, también se considera una capa de almacenamiento en el presente experimento, explicada en el siguiente subsección.

2.3.7. Almacenamiento de datos

Esta etapa esta enfocada en el almacenamiento de los datos intermedios generadas en el proceso de análisis de influencia social.

Para la etapa de recolección de datos utilizamos MongoDB [46] por la naturaleza del dato estudiado, el tweet. El Streaming API de Twitter recolecta los tweets en formato JSON (JavaScript Object Notation) y el número de atributos varia de acuerdo a la disponibilidad de sus valores, es decir, los tweets recolectados no tienen una estructura fija, ver Figura 2.3. Por lo tanto, una base de datos NoSQL orientada a documentos como MongoDB nos permite almacenar los tweets de forma rápida y fácil.

```
"id" : Long,
"source" : String,
"created_at" : String,
"text" : String,
"lang" : String,
"user" : {
    "id" : Long,
    "created_at" : String,
    "followers_count" : Integer,
    "friends_count" : Integer
}
```

```
"id" : Long,
"source" : String,
"created_at" : String,
"text" : String,
"lang" : String,
"user" : {
    "id" : Long,
    "created_at" : String,
    "followers_count" : Integer,
    "friends_count" : Integer
},
"retweetedStatus": {
    "id" : Long,
    "source" : String,
    "created_at" : String,
    "text" : String,
    "lang" : String,
    "user" : {
        "id" : Long,
        "created_at" : String,
        "followers_count" : Integer,
        "friends_count" : Integer
    }
}
```

FIGURA 2.3: Comparación de estructuras JSON de los tweets.
Fuente: Elaboración propia

En este punto es natural considerar el uso de base de datos orientada a grafos, como JanusGraph [47] o Neo4j. Sin embargo, tales herramientas no presentan una integración completa con el componente GraphX de Apache Spark.

2.4. Modelo de procedencia de datos

El concepto de procedencia es un historial de entidades, actividades y personas involucradas en la producción de piezas de datos, el cual permite evaluar su calidad y veracidad. Este tipo de datos se encuentran representados por una conjunto de estándares, definidos por el W3C Provenance Working Group. En el contexto de este trabajo, utilizaremos el modelo de procedencia de datos (PROV-DM, por sus siglas en ingles) [16].

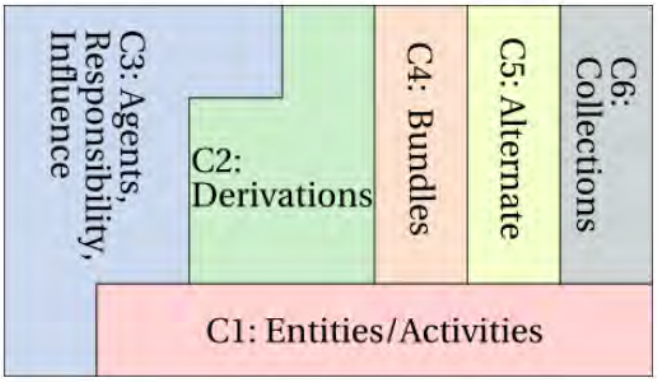


FIGURA 2.4: Componentes del PROV-DM.
Fuente: W3C Provenance Working Group [16]

El PROV-DM representa la procedencia con 6 componentes como se presenta en la Figura 2.4. No obstante, en este trabajo se limita al uso de los componentes: Entidades (C1), Derivaciones (C2) y Agentes (C3) para el análisis de redes sociales. En PROV-DM, las entidades son definidas como objetos físicos o digitales sin comportamiento inteligente. La derivación es la transformación de una entidad a otra entidad. Y un agente corresponde a una persona, organización u software con responsabilidad con otro agente, entidad o actividad. En nuestro trabajo, las entidades son representadas por los tweets, las derivaciones con la creación de nuevos tweets debido a una cita, copia de otro tweet y los agentes son representados por los usuarios que crean los tweets.

Adicionalmente, para mapear una red social con los componentes de PROV-DM, es también necesario definir los tipos de tweets generados por una derivación. En términos generales, definimos tres tipos de tweets:

a) Mensaje original, denota un mensaje que no ha sido derivado de otro mensaje, sino su contenido ha sido creado directamente por un agente.

b) Mensaje copia, denota un mensaje que ha sido derivado de otro mensaje creado en el pasado y con un copia exacta su contenido. En nuestro contexto, este tipo de mensaje se genera al retuitear (actividad) otro tweet (original, copia o modificado) realizado por un agente para expresar que comparte su opinión; ya sea original, copia o modificado.

c) Mensaje cita, denota un mensaje que ha sido derivado de otro mensaje creado en el pasado con o ninguna modificación en su contenido. En nuestro contexto, un agente emite este tipo de mensaje cuando cita (actividad) otro tweet para expresar que comparte o no su opinión; ya sea original, copia o modificado.

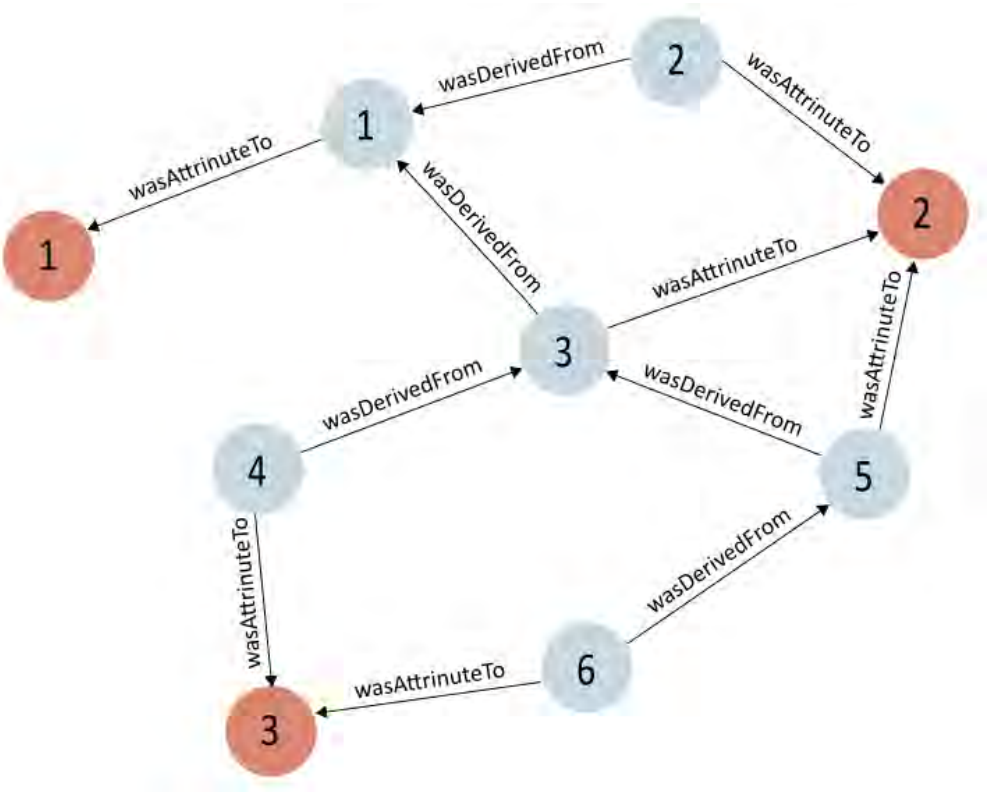


FIGURA 2.5: Ejemplo de mapeo de la red social utilizando el modelo PROV-DM.
Fuente: Elaboración propia

Este modelo es ampliamente utilizado en diferentes dominios. Jacob et al. [48], en el estudio de videojuegos, emplean el modelo para mapear los ciclos de juegos generados en las sesiones de los jugadores con el fin de identificar bucles malos que provocan el abandono del juego. En cambio, Corsar et al. [49] plantean el modelo para mapear la procedencia de las fuentes de datos externos del sistema de pasajeros en tiempo real, planteado para asegurar la calidad de la información proporciona a usuario objetivo. Ambos trabajos utilizan el modelo PROV-DM para estudiar la procedencia de sus datos de estudio.

Un ejemplo de su aplicación se muestra en la Figura 2.5, donde se muestra el resultado de mapear los datos de una red social en un grafo heterogéneo dirigido (dos o más tipos de nodos). De acuerdo al modelo PROV-DM, la relación de una entidad (tweet) y un agente (usuario) se mapea con un enlace de tipo *wasAttributeTo*; y la relación entre entidades, con un enlace de tipo *wasDerivedFrom*.

En nuestro trabajo, utilizaremos el modelo para mapear las relaciones de los datos recolectados desde Twitter. Para abstraer la complejidad de crear un grafo de propiedades dirigido heterogéneo, utilizaremos el componente GraphX de la plataforma de procesamiento distribuido de Apache Spark, la cual se explica en la siguiente sección.

2.5. Apache Spark

Con el gran número y rápido crecimiento de los datos, los métodos y técnicas tradicionales de aprendizaje de máquinas requieren adaptarse al paradigma de Big Data para analizar grandes volúmenes de datos. Por tal motivo, surgen diversas herramientas en cada una de las capas de Big Data [50], entre los cuales tenemos a Apache Hadoop y Apache Spark.

Apache Spark es un marco de trabajo de procesamiento distribuido de código libre desarrollado por la Universidad de California, Berkely AMPLab. Apache Spark mantiene la escalabilidad lineal y la tolerancia de fallas de MapReduce, pero extiende tres aspectos importantes. En primer lugar, tiene un motor avanzado Gráfico Acíclico Dirigido (DAG, por sus siglas en inglés). El motor optimiza el plan de ejecución de las operaciones, dividiéndolos en etapas de tareas en el programador DAG, el cual canaliza operaciones similares. En segundo lugar, habilita un conjunto de transformaciones que facilitan el preprocesamiento de los datos. En tercer lugar, Spark procesa los datos en memoria a través de los Resilient Distribuited Datasets (RDD, por sus siglas en inglés). Un RDD es una abstracción de almacenamiento que permite partir y distribuir los datos a través de un grupo de computadoras para realizar tareas en paralelo, ejecutando operaciones hasta 100 veces más rápido que Apache Hadoop.

En su versión 2.2.1, Spark cuenta con 4 componentes: Spark SQL, Spark Streaming, MLlib y GraphX, como se muestra en la Figura 2.6, empleadas en diferentes capas de Big Data. En particular, utilizamos GraphX como marco de procesamiento paralelo de grafos para analizar grandes redes sociales.

GraphX es un sistema de computo de grafos que se ejecuta encima del marco de procesamiento paralelo de datos de Spark, proporcionando así un único entorno de trabajo

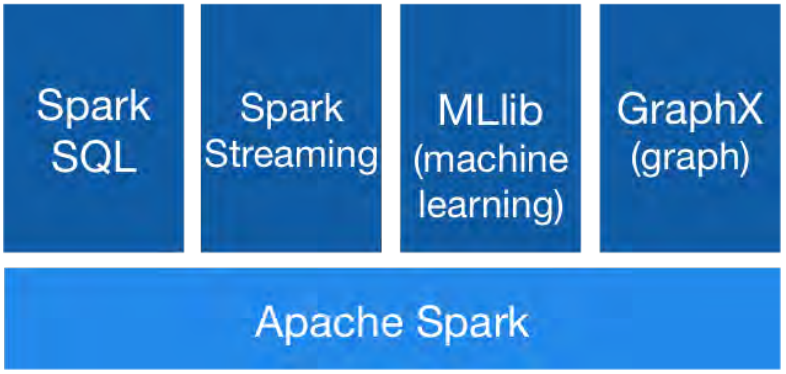


FIGURA 2.6: Componentes de Apache Spark.
Fuente: https://spark.apache.org

para realizar las tareas de pre-procesamiento, construcción e implementación de algoritmos de grafos para el análisis de redes sociales, a diferencia de otros marcos enfocados solamente en el diseño, implementación y aplicación de algoritmos de grafos [51].

GraphX proporciona el objeto Graph para construir un grafo, mediante dos RDDs: VertexRDD y EdgeRDD. VertexRDD almacena las propiedades de los nodos con un identificador único (ID). EdgeRDD almacena las propiedades de los enlaces junto con sus identificadores de nodo origen y nodo destino, como se muestra en la Figura 2.7. Mediante esta representación, GraphX admite más operaciones de cálculo sobre grafos que otros sistemas de procesamiento de grafos no admiten fácilmente, como agregar nuevas propiedades a los nodos, así como analizar resultados de cálculo de grafos como parte de unas operaciones como mapeado, agrupación o transformación.

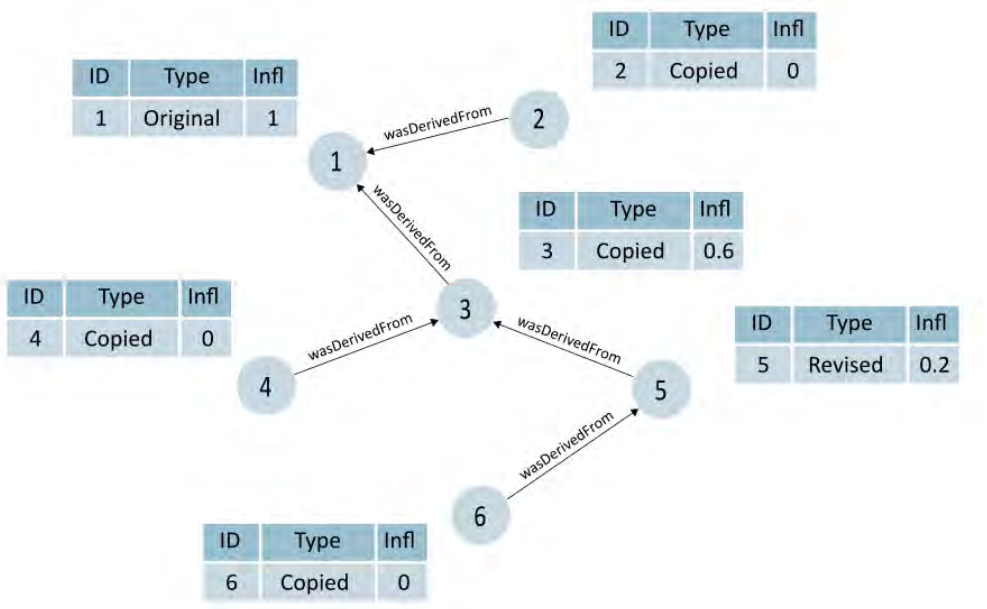


FIGURA 2.7: VertexRDD y EdgeRRD almacenan los datos de un grafo en Graphx.
Fuente: Elaboración propia

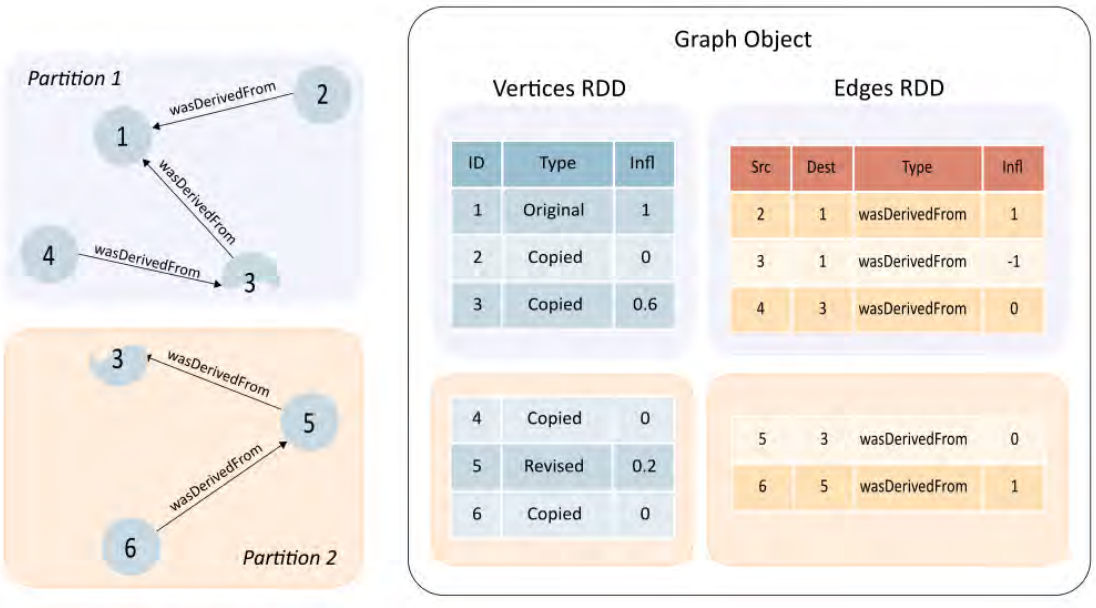


FIGURA 2.8: Representación tabular de la partición Vertex-Cut in GraphX.
Fuente: Elaboración propia

Como sistema de computo de grafos distribuido, GraphX proporciona un particionamiento y almacenamiento eficiente de grafos a través la estrategia de partición por corte de vértice. Debido a que los grafos del mundo real tienen más enlaces que vértices, es más eficiente mover los atributos de los vértices hacia los enlaces.

La partición por corte de vértice (vertex-cuts, por su traducción en inglés) asignan enlaces de manera uniforme a los máquinas y permiten que los vértices abarquen varias de ellas con el fin de asegurar el cálculo equilibrado. En la Figura 2.8 se muestra el corte de vértice para un mismo grafo. Se divide el grafo en dos particiones virtuales: Partición 1, Partición 2. Dentro del objeto Graph, EdgeRDD se divide con un identificador de partición (PID, por sus siglas en inglés) asociado a cada registro. VertexRDD se divide con PID asociado a cada registro y codifica la asignación del ID del vértice a las particiones de EdgeRDD que contienen los enlaces adyacentes.

Finalmente, GraphX permite la creación y adaptación de algoritmos para aprovechar el paradigma del procesamiento paralelo, a través de Pregel API. Pregel API proporciona una abstracción de mensajería paralela asíncrona (BSP, por sus siglas en inglés) restringida a la topología de los grafos [52].

Dada las características y abstracciones mencionadas, en el trabajo optamos por utilizar Spark y GraphX en las etapas de procesamiento, modelado, almacenamiento y análisis de redes sociales.

2.6. Análisis de sentimientos con VADER

Es un modelo de análisis de sentimientos de texto basado en reglas lexicales. VADER evalúa el sentimiento de cada palabra de un texto existente en el léxico usado por VADER, para calcular el sentimiento expresado en el texto. Por cada texto analizado, VADER produce 4 métricas de sentimiento, ver Ecuación 2.6. Las tres primeras métricas miden que tan *positivo*, *neutral* y *negativo* es un texto, y toman valores decimales entre 0 y 1. La última métrica es *compuesto*, conocida como puntuación compuesta pondera normaliza, que es la combinación de las tres primeras métricas, normalizado para tomar valores entre -1 (muy negativo) y 1 (muy positivo).

$$VADER(m) = [positivo, neutral, negativo, compuesto] \tag{2.6}$$

Para el cálculo de las métricas, VADER utiliza una lista de características léxicas que incluyen jergas (por ejemplo, *wu*), emoticones (por ejemplo, *:) :L*), emojis codificados en utf-8 e iniciales y acrónimos (por ejemplo, *lol*). Esto, permite a VADER obtener resultados excepcionales en la evaluación del contenido compartido en redes sociales, superando las calificaciones realizadas por intervención humana [53]. En el presente trabajo se utilizará la métrica compound para medir con un solo valor el sentimiento expresado en el contenido de los tweets.

Capítulo 3

Revisión del estado del arte

Las redes sociales en línea y el análisis de influencia son áreas de investigación populares centrados en el estudio del rol de los usuarios en el proceso de difusión de información. La habilidad de influir a otros usuarios también puede estudiarse utilizando otras áreas de trabajo relacionadas.

En este capítulo se describen trabajos recientes orientados a realizar la identificación de usuarios influyentes mediante algoritmos híbridos. Además, se indican el enfoque utilizado, los atributos seleccionados y los resultados obtenidos.

Dependiendo de la red social, la definición de influencia difiere. Por ejemplo, Peng et al.[54] estudian la influencia utilizando un red social, representada por un grafo G(V,E,W) dirigido, construido a partir de 20 millones de mensajes SMS/MMS generados por 0,4 millones de usuarios durante un periodo de tres semanas en Octubre 2012 de una de las redes celulares más grandes de China. En el grafo de la Figura 3.1, Peng et al. utilizan los nodos (V) para representar a los usuarios, los enlaces (E) representa un envío de SMS entre dos usuarios móviles, y el peso del enlace (W) es el número menor de SMS enviados entre los usuarios. En base al grafo no dirigido, Peng et al. proponen su algoritmo basado en entropía de información.

El algoritmo calcula la influencia en base a la influencia directa e indirecta. La influencia directa DI_i de un nodo i se calcula como la suma de la entropía de los SMS enviados entre el nodo i y los nodos que reciben un SMS del nodo i. En cambio, la influencia indirecta II_{ik} se calcula como la suma del producto de la influencia directa de los nodos intermedios existentes entre la relación de profundidad 1 entre el nodo i y un nodo k. Por ejemplo del grafo de la Figura 3.1, la influencia total del nodo A se obtiene de la suma de la influencia directa DI_{AB}, DI_{AD} y DI_{AC} con la suma de la influencia indirecta

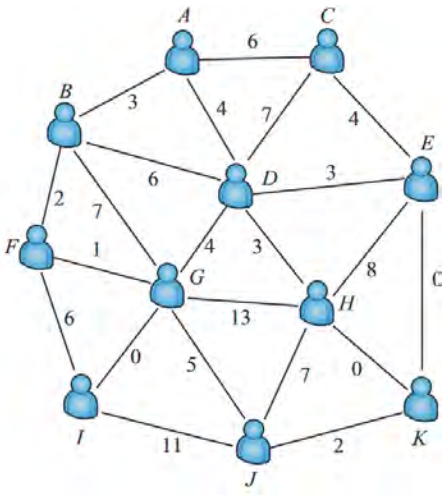


FIGURA 3.1: Grafo no dirigido ponderado basado en el número de interacciones efectivas entre dos nodos.
Fuente: Sancheng Peng et al. [54].

II_{AF}, II_{AG} y II_{AE}; de los nodos con la cual tiene un relación de profundidad 1. Para proteger la privacidad de los usuarios, Peng et al. remueven el contenido de los mensajes SMS y reemplaza el identificador único de los teléfonos celulares por un pseudocódigo. Finalmente, Peng et al. utilizan el modelo de epidemia como modelo de propagación de influencia para evaluar su algoritmo basado en entropía de información con los algoritmos Degree y Random, a través de la difusión de influencia. Por cada algoritmo, Peng et al. obtienen tres conjuntos de usuarios influyentes. Para cada conjunto de nodos se calcula el número de nodos influenciados por los nodos influyentes, obteniendo el algoritmo basado en entropía de información una mayor difusión de influencia, ver Figura 3.2.

Otras de las áreas empleadas para identificar usuarios influyentes es el aprendizaje de reglas de asociación propuesto por Borg et al.[25]. Los autores capturan datos de 195 grupos y páginas públicas de Facebook, la cuál consiste en más de 56 millones de post, 560 millones de comentarios, 7,3 millones de *like* y 820 millones de usuarios. A partir de los datos, se genera 46 170 reglas con una confianza mayor al 95 % para los datos de la página de Facebook *OccupyTogether*, con una exactitud de 0,886 y una precisión de 0,291. Los usuarios influyentes son clasificados según la frecuencia con la que aparecen en las reglas generadas. Para validar sus resultados, los autores lo compara con la lista de usuarios influyentes obtenidos con los métodos de centralidad PageRank y la centralidad de grado, como se aprecia en la Tabla 3.1. De acuerdo a la tabla, del 50 % de los usuarios influyentes existente en la página de *OccupyTogether*, el 95 % los usuarios influyentes identificados con los métodos de centralidad de grado y PageRank son similares. Por otro, los usuarios influyentes obtenidos por el método basado en reglas de

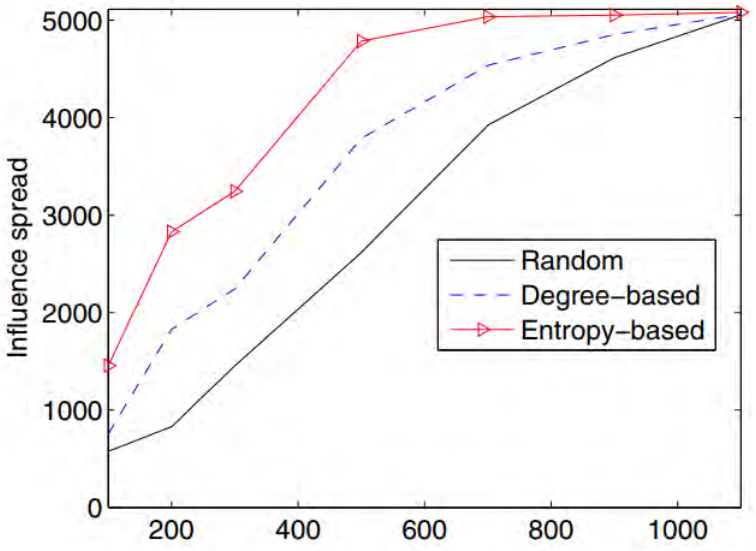


FIGURA 3.2: Comparación de difusión de influencia de los algoritmos Random, Degree y Entropy con diferentes k = (100, 200, 300, 500, 700, 900, 1100) nodos influenciadores (líderes de opinión).
Fuente: Peng et al. [54].

asociación (ASR, por sus siglas en inglés) presentan una semejanza de 51 % comparado con la centralidad de grado y del 53 % comparado con la centralidad PageRank.

Percent of Top Users	Users	Degree ∩ ASR	Page Rank ∩ ASR	Page Rank ∩ Degree
1 %	4	0.75	0.75	0.75
5 %	20	0.45	0.45	0.95
10 %	41	0.488	0.512	0.927
25 %	104	0.462	0.49	0.923
50 %	209	0.512	0.526	0.947
75 %	313	0.502	0.556	0.92
100 %	418	0.517	0.565	0.928

TABLA 3.1: Comparación de la similitud de usuarios influyentes para la página *OccupyTogether*.
Fuente: Borg et al. [25].

Utilizando otra perspectiva, Chen et al.[23] proponen el algoritmo OLMiner para buscar lideres de opinión (los usuarios más influyentes) en grandes redes sociales. Para ello, OLMiner divide el procesamiento en 5 partes: la verificación de granularidad de datos, la construcción de un grafo para representar la red social, la detección de comunidades de usuarios, la generación de candidatos a lideres de opinión y, finalmente, la selección de los líderes de opinión (ver Figura 3.3).

Chen et al. utilizan 6 928 artículos relacionados a la marca de carros LEXUS y 6 418, a

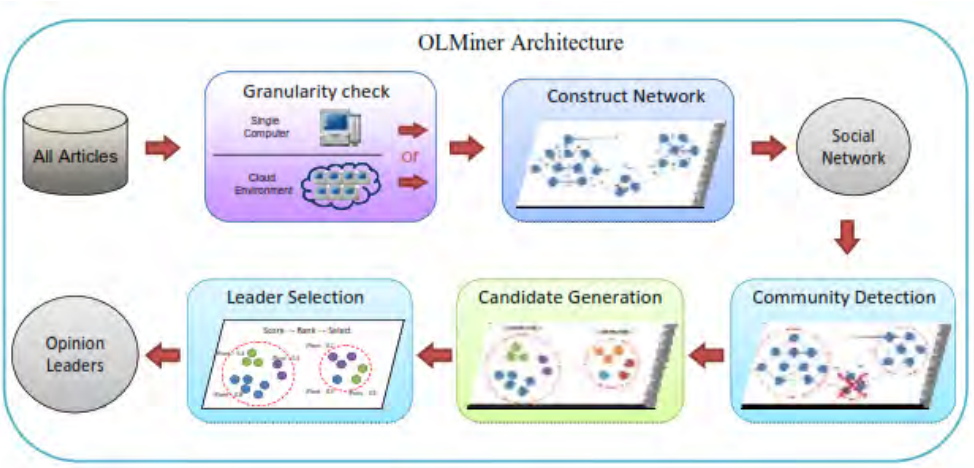


FIGURA 3.3: Marco de trabajo de OLMiner.
Fuente: Yi-Cheng Chen et al. [23].

la marca AUDI, un total de dos dataset recolectados del forum Mobile01[1]. Por cada dataset, Chen et al. construyen un grafo homogéneo dirigo *G(V,E,W)*, donde cada usuario es representado por un nodo $v \in V$, cada relación comentario_articulo entre dos usuarios v y u es representado con un enlace $e \in E$ y el peso de los enlaces se asigna a través del calculo de similaridad entre los nodos adyacentes u y v $(sim(u,v))$. Finalizado la construcción del grafo, utiliza el algoritmo de agrupación H_clustering [55] utilizando los atributos para centralizar el trabajo de identificación dentro de comunidades significativas. Para identificar una comunidad significativa, Chen et al. utilizan la Ecuación 3.1, donde una comunidad c_j es significativa $(c_j \in C)$, si el número de nodos de la comunidad n_j es mayor o igual a la proporción entre la cantidad de nodos de todas las comunidades y el k número de lideres de opinión deseados.

$$C_s = \{c_j \in C | n_j \geq \sum_{i=1}^{p} \frac{n_i}{k}\} \tag{3.1}$$

Para generar los candidatos a lideres de opinión, Chen et al. emplean el algoritmo de agrupación k-means [56] utilizando las características de cada nodo usuario: total de artículos publicados, la probabilidad de que otro usuario comente un articulo, el grado de experiencia en el tópico y la probabilidad de responde artículos de otros usuarios. A cada grupo obtenido ck_{ij} se calcula el score de agrupación $score(ck_{ij})$, considerando los 4 atributos mencionados. A partir del score, se seleccionan los líderes de opinión. Para evaluar los resultados obtenidos, Chen et al. utilizan la métrica de propagación de influencia en los lideres de opinión identificados por el algoritmo OLMiner y el algoritmo de agrupación *att_clustering*. En la Figura 3.4 se observa que los k líderes de opinión obtenidos por el algoritmo OLMiner logran influenciar a una mayor cantidad de nodos

[1]Mobile01: https://www.mobile01.com/

en el proceso de propagación de la difusión, en comparación a los líderes de líderes de opinión identificados por el algoritmo *att_clustering*.

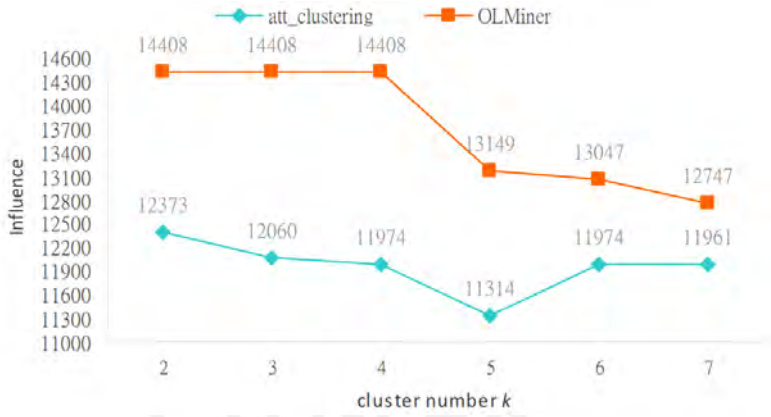


FIGURA 3.4: Propagación de influencia generada por los algoritmos OLMiner (línea naranja) y att_clustering (línea celeste) con variaciones en el número de k líderes de opinión.
Fuente: Chen et al. [23].

Utilizando el mismo enfoque, Jia et al. [1] plantean un esquema de identificación de usuarios influyentes a través de tres pasos: la caracterización de atributos estáticos de los usuarios, la generación de grupos centralizados de usuarios utilizando el algoritmo K-means [56] y, por último, la aplicación del algoritmo PageRank [57] en cada grupo de usuarios. Jia et al. utilizan datos de telefonía celular de una región de China. No indicando el volumen de los datos, Jia et al. dividen los datos de los usuarios en tres categorías: datos de identidad (índice, ciudad, nivel de servicio, indicador de usuario contratado, edad y género), datos de consumo (cuota mensual, tipo de paquete, duración de llamada, volumen de tráfico, precio de celular y tipo de servicio) y datos de preferencias (marca de celular, modelo de celular, área de residencia, horario de atención preferido, aplicación preferida e interés del usuario). Creando un vector de características para cada usuario, se aplica el algoritmo de agrupación K-means para generar grupos de usuarios con alta correlación. Aplicando el algoritmo PageRank, ver Ecuación 3.2, la influencia de un usuario i ($Pr_{(u_i)}$) esta definido como la suma de la influencia de cada usuario j que recibió una llamada del un usuario i, todo divido por el número de llamadas que el usuario j realizo a otros usuarios del mismo grupo. N es el total de usuarios agrupados en el mismo cluster y d es el factor de atenuación del algoritmo.

$$Pr_{(u_i)} = \frac{1-d}{N} + d\left(\sum_{u_j \in In_{(u_i)}} \frac{Pr_{(u_j)}}{\mid Out_{(u_j)} \mid}\right) \tag{3.2}$$

Para validar su algoritmo, Jia et al. comparan el rendimiento entre una campaña de

publicidad tradicional y otra, utilizando el 50 % de los usuarios más influyentes obtenidos por el algoritmo. En la Figura 3.5 se aprecia que la tasa de las visitas a los productos es mayor al emplear los usuarios con alto poder de influencia en cada semana después de lanzamiento de la campaña publicitaria.

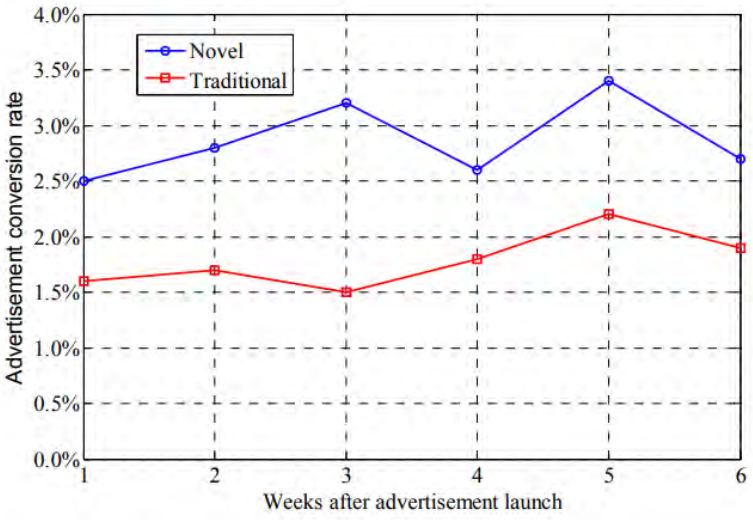


FIGURA 3.5: Tasa de conversión generada por una campaña publicitaria tradicional (rojo) y una, empleando el algoritmo propuesto (azul).
Fuente: Jia et al. [1].

Por otro parte, para Sun et al. [58] los líderes de opinión intercambian información en múltiples redes o relaciones al mismo tiempo. Por ende, los autores proponen un grafo homogéneo multi-relacional dirigido $G(V, E, R, F)$ para identificar a los usuarios más influyentes considerando los múltiples enlaces $\in E$ existentes entre dos nodos $\in V$. Además, R es el vector de relaciones existente entre los nodos y F es el mapeo entre los enlaces $\in E$ que conforman cada vector de relaciones $\in R$. Un ejemplo de la estructura propuesta se observa en la Figura 3.6, donde cada usuarios comparte dos tipos de relaciones (r_1, r_0) por cada enlace. El símbolo $\oslash$ indica la inexistencia de una relación entre los nodos. A partir del grafo, se construye una matriz de importancia de D a partir de la matriz de adyacencia del grafo y los vector R. Se aplica la multiplicación iterativa de matriz R y D para simular el proceso de señalización (planteado por Sun et al.) para finalmente, calcular el grado de importancia de los nodos usuarios.

Para evaluar el algoritmo, se obtiene los datos generados por 30 453 usuarios de la red social Douban.com[2] durante un periodo de dos días. A partir de los datos, construye un grafo homogéneo multi-relacional dirigido de 30 453 nodos, 55 865 enlaces y 1 789 567 comentarios de libros. Sun et al. comparan su algoritmo con los algoritmos PageRank [57]

[2]Douban: https://www.douban.com/

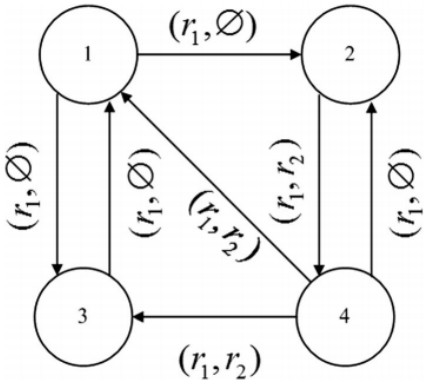


FIGURA 3.6: Grafo multi-relacional de 4 nodos usuarios.
Fuente: Sun et al. [58].

y HITS [59]. Además, realiza una votación manual de líderes de opinión para comparar la exactitud de los tres algoritmos evaluados.

	Manual voting	Our proposed algorithm	PageRank algorithm	HITS algorithm
1	No.701	No.701	No.6939	No.6939
2	No.7018	No.7018	No.3303	No.3303
3	No.174	No.1239	No.701	No.701
4	No.3377	No.3356	No.8220	No.8220
5	No.3356	No.174	No.7018	No.7018
6	No.7132	No.209	No.1239	No.3549
7	No.4589	No.3549	No.3549	No.1239
8	No.209	No.7132	No.286	No.286
9	No.2828	No.3561	No.3356	No.3356
10	No.6461	No.4323	No.7911	No.2914
11	No.3303	No.3303	No.174	No.174
12	No.3549	No.702	No.13237	No.13237
13	No.6939	No.2828	No.286	No.286
14	No.4323	No.6461	No.702	No.3561
15	No.8220	No.4589	No.209	No.209
16	No.702	No.6939	No.7132	No.702
17	No.4113	No.2914	No.3561	No.2828
18	No. 377	No.8220	No.2828	No.7132
19	No.7911	No.13237	No.2914	No.7911
20	No.13237	No.7911	No.6395	No.6395

TABLA 3.2: Comparación de los resultados obtenidos de los algoritmos PageRank, HITS y el propuesto por Sun et al. con una votación manual.
Fuente: Sun et al. [58].

De acuerdo a la Tabla 3.2, la lista de los veinte líderes de opinión más influyentes obtenidos por el algoritmo propuesto son son más cercanos a los resultados obtenidos en la votación manual de líderes de opinión, logrando mayor exactitud que los algoritmos PageRank y HITS.

Considerando el procesamiento de grandes volúmenes de datos, Zhou et al.[7] elaboran un algoritmo distribuido utilizando Apache Spark para medir la influencia de los usuarios

de la red social Sina Weibo. Zhou et al. emplean datos de 1 262 518 usuarios y 114 286 565 mensajes obtenidos durante un periodo de dos años, Abril 2013 - Marzo 2015. Los datos son etiquetados con el valor de influencia calculado a partir del número de reacción que recibe un usuarios por sus mensajes. Por ejemplo, el usuario u logra 13, 8, 7, 5 y 1 reacciones con sus mensajes A, B, C, D y E, respectivamente, por lo tanto el valor de su influencia es 4 porque el número de reacciones baja en su quinto mensaje. De manera similar, el usuario j tiene una influencia de 3 porque las reacciones a sus mensajes bajan en su cuarto mensaje, ver Tabla 3.3.

User	**A**	**B**	**C**	**D**	**E**	**User influence score**
$user_i$	13	8	7	5	1	4
$user_j$	12	9	3	1	0	3

TABLA 3.3: Cálculo de la influencia de los usuarios de Sino Weibo.
Fuente: Zhou et al.[7]

Cada usuario es etiquetado con un valor de acuerdo a su influencia. Para Zhou et al., la influencia de cada usuario se define por sus propiedades y sus actividades pasadas, las cuales definen a cada usuario como un conjunto de trece características estadísticas y seis características de tópico. De los datos, se obtiene como características estadísticas del usuario: el número de seguidores, el número de amigos, el número de favoritos, el número de usuarios que se siguen mutuamente, el número de mensajes (enviados, reenviados), el número de amigos importantes, un indicador si existe una dirección URL en las propiedades del usuario, un indicador si el usuario ha sido verificado por Sina Weibo[3], el tamaño del nombre de usuario, el tamaño de descripción, la fecha de registro del usuario, el género del usuario y el ratio del número de seguidores con el número de amigos. Además, de los mensajes se obtienen seis características estadísticas: el número de hashtags, enlaces, menciones y palabras del mensajes, imágenes y el tamaño del mensaje.

Para obtener las características del tópico, Zhou et al. proponen el algoritmo de fusión de frases para juntar el contenido de todos los mensajes de un usuario en un único documento. Luego, aplicando el algoritmo Latent Dirichlet Allocation [60] (LDA por sus siglas en inglés) todos los documentos son utilizados para identificar tópicos de conversación y la distribución del interés de los usuarios en cada tópico. Las trece características estadísticas de los usuarios junto con las seis de los mensajes son utilizados como atributos para generar los modelos de regresión Random Forest, Decision Tree y Gradient Boosting, implementados en el componente MLlib de Apache Spark [61], utilizando como etiqueta la influencia de los usuarios previamente calculados y el 80 % de los datos

[3]Sina Weibo: https://www.weibo.com

para entrenamiento y el 20 % restante para la validación los resultados. Como resultado, Jun Zhou demuestra que la incorporación de las características de los tópicos de los mensajes mejoran en un 24 % la precisión los modelos utilizados para identificar a los usuarios influyentes.

Capítulo 4

Procedimiento para identificar los líderes de opinión

El objetivo del procedimiento descrito en este capítulo es cuantificar la influencia que genera los usuarios dentro de sus redes sociales. Para esto, se plantea utilizar un algoritmo híbrido basado en el estudio de los atributos estáticos y los atributos de interacción de los usuarios. Por un lado, los atributos de interacción no están de forma explícita entre los datos recolectados de Twitter, por lo que es necesario un proceso de construcción de redes sociales. Por otro lado, para proteger la privacidad de los usuarios se requiere un proceso de sanatización y preprocesamiento de los atributos sensibles.

Con este fin, el procedimiento se ha divido en siete etapas de manera análoga a lo explicado en la Sección 2.3. La primera es la extracción de tweets desde Twitter. Una vez obtenido los tweets, pasan por una etapa de preprocesamiento de datos, se realiza la sanitización de atributos sensibles y, luego, se realiza un análisis univariante para la selección de atributos. Una vez que se tengan seleccionado los atributos, se aplica el algoritmo de construcción de red social basado en el modelo procedencia de datos y aplicando el método de red completa. El proceso de análisis de influencia social, esta dividida a su vez en dos etapas: la primera es excluir a los usuarios con poca participación en la generación de interacción con un algoritmo de detección de comunidades. Una vez que las comunidades con pocos miembros de usuarios sean excluidas, se aplica el algoritmo híbrido propuesto para cuantificar la influencia que ejerce cada usuario desde comunidad. Finalmente, del resultado del algoritmo híbrido se selecciona a los líderes de opinión.

4.1. Extracción de tweets desde Twitter

El procedimiento de extracción de tweet es una tarea completamente de programación, primero se realiza una integración con API de Streaming de Twitter a través la librería Twitter4j, programada en el lenguaje de programación Java. La Figura 4.1 y el Algoritmo 1 dan el esquema y los pasos del procedimiento seguido. Para conectar al API de Streaming de Twitter, la librería Twitter4j requiere la creación de un objeto TwitterStream con las credenciales de autenticación *oAuth*, generadas en el portal de administración de aplicaciones de Twitter[1]. La extracción de tweets es responsabilidad del objeto StatusListener, y se crea con las palabras claves definidas en la lista T para filtrar los tweets. Luego, se crea una cadena de conexión a MongoDB con las credenciales de conexión de *Conn* para almacenar los tweets extraídos (lineas 1-3, Algoritmo 1).

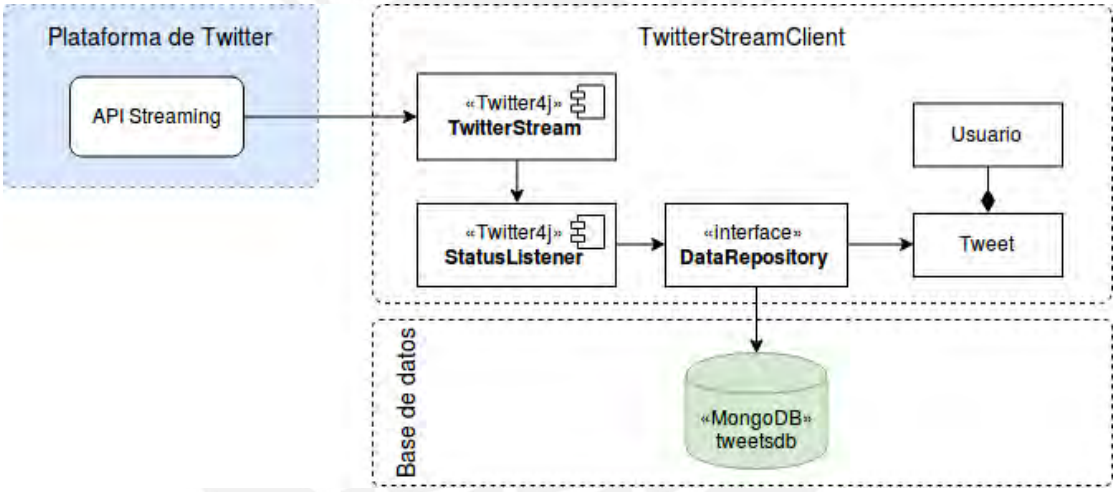


FIGURA 4.1: Proceso de extracción de tweets a través de API Streaming de Twitter. Fuente: Elaboración propia.

El objeto *sListener* establece una conexión continua con Twitter para extraer en casi tiempo real los tweets en formato JSON, *tweetjson*, que tengan en su contenido una de las palabras claves definidas en la lista T (líneas 1-3, Algoritmo 1). Se extraen los atributos de *tweetjson* para crear los objetos *usuario* y *tweet*, respectivamente. Los nuevos objetos creados son almacenados en la base de datos *tweetsDB* utilizando el objeto *dRepository* (lineas 4-9, Algoritmo 1).

Para la ejecución del Algoritmo 1, definimos como palabras claves una lista T con la frase "calentamiento global" traducido en trece idiomas diferentes, seleccionados de acuerdo al idioma oficial de las potencias mundiales según el Producto Bruto Interno[2] (PBI,

[1]Twitter Application Management: https://apps.twitter.com/

[2]World Economic Outlook Database: http://statisticstimes.com/economy/countries-by-projected-gdp.php

según sus siglas en inglés). La recolección de tweets inicio el 03/07/2018 y finalizo el 23/09/2018. En los 82 días de recolección se obtuvieron 1 589 131 objetos JSON, cada objeto contiene entre uno a tres objetos *Tweet* y *User*. En total, se obtienen 953 692 objetos *User* y 1 620 314 objetos *Tweet* desde los objetos JSON procesados.

Algoritmo 1: Extracción de tweets - API Streaming Twitter

Input : T: lista de las palabras claves sobre el "Calentamiento Global";
oAuth: lista de credenciales de Twitter;
Conn: lista de credenciales a la base de datos MongoDB
Output: tweetsDB: base de datos tweets

```
tStream ← crear el objeto TwitterStream con las credenciales de oAuth;
sListener ← crear el objeto StatusListener con las palabras de T;
dRepository ← crear el objeto DataRepository con las credenciales de Conn;
mientras sListener hacer
    tweetjson ← extraer un tweet publicado con las palabras de T;
    usuario ← extraer los atributos del objeto Usuario de tweetjson;
    tweet ← extraer los atributos del objeto tweet de tweetjson;
    dRepository.save(tweet)
fin
```

Output: tweetsDB

Si bien, el presente experimento se enfoca en recolectar tweets creados o compartidos con respecto al tema del calentamiento global, el algoritmo puede replicarse en otros contextos. Una vez que tenemos los tweets almacenados, podemos uniformizarlos.

4.2. Preprocesamiento de tweets

A partir de los tweets almacenados se realiza un análisis de frecuencia de la variable *lang* que representa el idioma del contenido del tweet, revisar Apéndice A. En la Tabla 4.1 se muestra el resultado obtenido del análisis de frecuencias y evidencia que los tweets escritos en ingles representan el 82.29 % del total tweet recolectados, le siguen los tweets en español (6.43 %), portugués (5.65 %), francés (3.06 %), indefinido (1.42 %) y en otros idiomas (3.14 %), respectivamente. Por lo tanto, los tweets en inglés son seleccionados como nuestra población de estudio por su predominancia en la generación de contenido relacionado al calentamiento global.

Definido la población de tweets en idioma inglés, los objetos *User* y *Tweet* existentes pasan por el proceso de selección de atributos, también conocida como selección de características o selección de variables, y la selección de instancias. Para la selección de atributos, se realiza un análisis estadístico de los valores de los atributos en las instancias

Idioma	Código BCP 47	Frecuencia Absoluta	Frecuencia Relativa
English	en	1 300 981	80.29 %
Spanish	fr	104 194	6.43 %
Portuguese	es	91 586	5.65 %
French	pt	49 626	3.06 %
Others		50 933	3.14 %
Undefined		22 994	1.42 %
Total		1 620 314	100.0 %

TABLA 4.1: Distribución de frecuencia del atributo *lang* del objeto *Tweet*.
Fuente: Elaboración Propia.

de los objetos *User* y *Tweet*. El resultado del análisis para los atributos del objeto *User* se muestra en la Tabla 4.2-A, y para el objeto *Tweet*, en la Tabla 4.2-B.

Atributos	% Valores Faltantes
id	0 %
name	4.8 %
created_at	0 %
location	32.63 %
followers_count	0.69 %
friends_count	0.51 %

(A)

Atributos	% Valores Faltantes
id	0 %
source	0 %
created_at	0 %
text	0 %
lang	0 %
retweet_count	95.09 %
favorite_count	95.91 %
in_reply_to_user_id	91.09 %
in_reply_to_status_id	90.36 %
quoted_status_id	70.22 %
current_user_retweet	100 %

(B)

TABLA 4.2: Estadístico de valores faltantes en las instancias de los objetos *User* (A) y los objetos *Tweet* (B).
Fuente: Elaboración propia.

De acuerdo a la Tabla 4.2, no se considerará el atributo *location* del objeto usuario porque no es significante para nuestro trabajo. En el objeto tweet no se considerán los atributos *retweet_count*, *favorite_count*, *in_reply_to_user_id*, *current_user_retweet*, *in_reply_to_status_id* y *quoted_status_id* del objeto *Tweet* porque aproximadamente el 90 % de los valores registrados en sus instancias son vacíos o nulos, por consiguiente, en el proceso de selección de atributos se etiquetan como atributos eliminados. Los atributos *id*, *created_at*, *name*, *followers_count* y *friends_count* se etiquetan como atributos de estudio.

Con respecto a selección de instancias o registro, ninguna instancia *Tweet* y *User* se eliminan. Los atributos de estudio con valores nulos o vacíos, como por ejemplo *followers_count* y *friends_count* del objeto *User*, se completan con el valor mínimo aceptado dependiendo del tipo de datos almacenado.

Para proteger los valores de los atributos sensibles de los objetos *User* y *Tweet*, se emplea el algoritmo de Murmurhash3 [62]. El algoritmo se aplicará en los atributos *id* del objeto *User* y los atributos *id* y *text* del objeto *Tweet*. Los atributos identificadores (*id*) son atributos muy sensibles porque permite la identificación los propietarios de los tweets; sin embargo, son necesarios para identificar a los líderes de opinión dentro de la población de estudio.

Al final de esta etapa, los atributos *id*, *created_at*, *followers_count* y *friends_count* del objeto *User*, y los atributos *id*, *source*, *created_at*, *text* y *lang* del objeto *Tweet* son nuestros atributos de estudio y servirán de entrada en la elaboración de la red social, detallado en la siguiente sección.

4.3. Construcción de la red social

Para la construcción de la red social se utilizó el lenguaje de programación funcional Scala [63] en su versión 2.11 en conjunto con el framework de procesamiento paralelo de grafos GraphX [64].

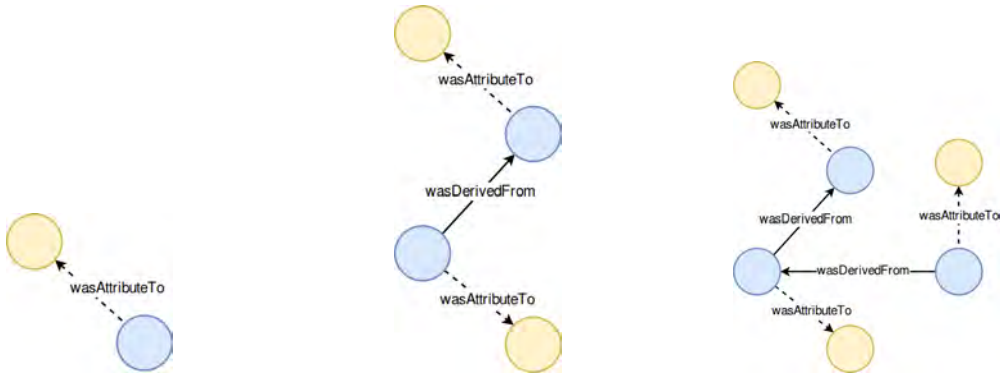
En el procesamiento de los objetos JSON se identifican tres tipos de estructuras: tweet original, tweet generado copia y tweet generado por cita, ver Figura 4.2. Un tweet es generado por copia si tiene el atributo *retweetedStatus*, un tweet es generado por cita tiene el atributo *quotedStatus* y un tweet es original si no cuenta con ningún atributo anteriormente mencionado. Utilizando los atributos *retweetedStatus* y *quotedStatus* se definen tres formas de interacción: tweet original, tweet copia y tweet cita. La Figura 4.3 muestran las estructuras de las interacciones utilizando la nomenclatura del modelo PROV-DM. En ella, el enlace *WasAttributeTo* representa la creación de un objeto *Tweet* por parte de un objeto *User*. La cita o copia de un objeto *Tweet* a partir de otro objeto *Tweet* se representa con el enlace *WasDerivedFrom*, no existiendo una relación directa entre dos objetos *User*.
De acuerdo al Algoritmo 2, los objetos *User* y *Tweet* pasan por un proceso de transformación de acuerdo al tipo de nodo:

- Para crear un nodo *User* a partir de un objeto *User*: el valor del atributo *id* pasa por la función Murmurhash3 para generar un valor numérico, ocultando la identidad de los usuarios de Twitter. Por otra parte, se crea el atributo *created_at* para almacenar la fecha de creación de la cuenta del usuario y el atributo *influence_score* para almacenar el valor numérico de la influencia ejercida por el usuario.

FIGURA 4.2: Las estructuras JSON de los tweets: original (A), copia o retweet (B) y cita (C).
Fuente: Elaboración propia.

- Para crear un nodo *Tweet* a partir de un objeto *Tweet*: se aplica análisis de sentimiento al atributo *text* utilizando el método VADER. Se crea el atributo *sentiment_score* para almacenar el resultado del análisis de sentimiento, el atributo *created_at* para guardar la fecha de creación del tweet y el atributo *influence_score* para almacenar el valor numérico de la influencia ejercida por el tweet.

Por otro lado, en la creación de los enlaces *WasAttributeTo* y *WasDerivedTo* se realiza un proceso para asignar un peso w a la relación de dos nodos. Los enlaces *WasAttributeTo* se le asigna un peso de 0. En cambio, los pesos de los enlaces *WasDerivedTo* se calculan en base al valor del atributo *sentiment_score* (sc, por sus siglas en inglés) de los nodos *Tweet* enlazados aplicando la Ecuación 4.1.



(A) (B) (C)

FIGURA 4.3: Las estructuras de grafos de los tweets: tweet original (A), tweet copia (B) y tweet cita (C).
Fuente: Elaboración propia.

$$w_{ij} = \begin{cases} sc_i - sc_j & \text{, si } sc_i > sc_j \\ 1 & \text{, si } sc_i = sc_j \\ 1 + sc_j - sc_i & \text{, si } sc_i < sc_j \end{cases} \tag{4.1}$$

El peso de un enlace *WasDerivedTo*, $w_{ij} \in W$ entre los nodos *Tweet* i e j se define de acuerdo al valor de sus atributos sc_i e sc_j. Si el sc_i es mayor a sc_j, w, el peso w_{ij} es igual a $sc_i - sc_j$. Si sc_i e sc_j son iguales, el peso w_{ij} es igual a 1. Y si sc_j es mayor a sc_i, el peso w_{ij} es igual a $sc_j - sc_i$.

En el Algoritmo 2, cada objeto JSON es procesado de acuerdo a la estructura que presenta. Si *type_pattern* es igual a 700, se procesará un tweet original, donde se crean un nodo *User* y un nodo *Tweet* relacionados por un enlace *WasAttributeTo* con peso 0 (línea 5-11, Algoritmo 2). Si *type_pattern* es igual a 710, se procesará un tweet copia con dos pares de nodos *User* y *Tweet*, cada uno con una relación *WasAttributeTo* con peso 0 y una relación *WasDerivedFrom* entre los nodos *Tweet* t y t_r, con un peso asignado por la Ecuación 4.1 (líneas 12-23, Algoritmo 2). Si *type_pattern* es igual a 701, se procesará un tweet cita con dos pares de nodos *User* y *Tweet*, cada uno con una relación *WasAttributeTo* con peso 0 y una relación *WasDerivedFrom* entre los nodos *Tweet* t y t_q, con un peso asignado por la Ecuación 4.1 (líneas 25-36, Algoritmo 2). Por último, Si *type_pattern* es igual a 711, se procesará un tweet donde exista una copia y cita a la vez, con 3 pares de nodos *User* y *Tweet*, cada uno con una relación *WasAttributeTo* con peso 0 y dos relaciones *WasDerivedFrom*, una entre los nodos *Tweet* t y t_r y la otra entre los nodos *Tweet* t_q y t_r, con pesos asignados de acuerdo a la Ecuación 4.1 (líneas 37-54, Algoritmo 2). Como salida del algoritmo, se obtiene el grafo dirigido heterogéneo

$G(V, E, W)$. Los datos de los nodos y enlaces del grafo obtenido se almacenarán en el directorio HDFS */graph/t_vertex* y */graph/t_edge*, respectivamente, y estarán disponibles para la fase de detección de comunidades significativas, explicada en la siguiente sección.

Algoritmo 2: Construcción de red social.

Input : T: lista de registros de los tweet en formato JSON

Output: *G(V,E,W)*: una red social

```
1  V ← ∅; E ← ∅; W ← ∅;
2  para cada tweetjson ∈ T hacer
       /* Validar el patron del tweetjson                          */
3      type_pattern ← getTypePattern(tweetjson);
       /* Procesar un tweet original                               */
4      si type_pattern == 700 entonces
5          u ← getUserNode(tweetjson); t ← getTweetNode(tweetjson);
6          (t, u) ← getWasAttributeTo(tweetjson);
7          V ← V ∪ u, t;
8          E ← E ∪ (t, u);
9          W ← W ∪ 0;
10     fin
       /* Procesar un tweet copia                                  */
11     si type_pattern == 710 entonces
12         u ← getUserNode(tweetjson); t ← getTweetNode(tweetjson);
13         (t, u) ← getWasAttributeTo(tweetjson);
14         retweetJson ← tweetJson.get(retweetedStatus);
15         u_r ← getUserNode(retweetJson); t_r ← getTweetNode(retweetJson);
16         (t_r, u_r) ← getWasAttributeTo(retweetJson);
17         (t, t_r) ← getWasDerivedFrom(tweetjson, retweetJson);
18         V ← V ∪ u, u_r, t, t_r;
19         E ← E ∪ (t, u), (t_r, u_r), (t, t_r);
20         w_{t,t_r} ← peso asignado por la Ecuación 4.1;
21         W ← W ∪ w_{t,t_r};
22     fin
23 fin
```

```
24
    /* Procesar un tweet cita                                          */
25  si type_pattern == 701 entonces
26      u ← getUserNode(tweetjson); t ← getTweetNode(tweetjson);
27      (t, u) ← getWasAttributeTo(tweetjson);
28      quotedJson ← tweetJson.get(quotedStatus);
29      u_q ← getUserNode(quotedJson); t_q ← getTweetNode(quotedJson);
30      (t_q, u_q) ← getWasAttributeTo(quotedJson);
31      (t, t_q) ← getWasDerivedFrom(tweetjson, quotedJson);
32      V ← V ∪ u, u_q, t, t_q;
33      E ← E ∪ (t, u), (t_q, u_q), (t, t_q);
34      w_{t,t_q} ← peso asignado por la Ecuación 4.1;
35      W ← W ∪ w_{t,t_q};
36  fin
    /* Procesar un tweet copia, cita                                   */
37  si type_pattern == 711 entonces
38      u ← getUserNode(tweetjson); t ← getTweetNode(tweetjson);
39      (t, u) ← getWasAttributeTo(tweetjson);
40      retweetJson ← tweetJson.get(retweetedStatus);
41      u_r ← getUserNode(retweetJson); t_r ← getTweetNode(retweetJson);
42      (t_r, u_r) ← getWasAttributeTo(retweetJson);
43      quotedJson ← tweetJson.get(quotedStatus);
44      u_q ← getUserNode(quotedJson); t_q ← getTweetNode(quotedJson);
45      (t_q, u_q) ← getWasAttributeTo(quotedJson);
46      (t, t_r) ← getWasDerivedFrom(tweetjson, retweetJson);
47      (t_r, t_q) ← getWasDerivedFrom(retweetJson, quotedJson);
48      V ← V ∪ u, u_r, u_q, t, t_r, t_q;
49      E ← E ∪ (t, u), (t_r, u_r), (t_q, u_q), (t, t_r), (t_r, t_q);
50      w_{t,t_r} ← peso asignado por la Ecuación 4.1;
51      W ← W ∪ w_{t,t_r};
52      w_{t_q,t_r} ← peso asignado por la Ecuación 4.1;
53      W ← W ∪ w_{t_q,t_r};
54  fin
55 fin
```

Output: *G(V, E, W)*

Algoritmo 3: Detección de comunidades

Input : $G(V, E, W)$: una red social.
Output: C_s: conjunto de comunidades significativas

1 C $\leftarrow \varnothing$; $C_s \leftarrow \varnothing$;
2 graph $\leftarrow G(V, E, W)$;
`/* Algoritmo Componente Conectado */`
3 $G\_cc \leftarrow$ graph.connectedComponents();
4 $V_L \leftarrow$ extraer los vértices etiquetados de G_cc;
5 $list\_cc \leftarrow$ extraer las etiquetas únicas de V_L;
6 **para cada** $label\_cc_i \in list\_cc$ **hacer**
7 $v_i \leftarrow$ extraer los vértices $v \in V_L$ etiquetados con $label$;
8 $e_i \leftarrow$ extraer los enlaces $e(v_s, v_d) \in E$, donde $v_i = v_s$ ó $v_i = v_d$;
9 $w_i \leftarrow 0$;
10 C $\leftarrow$ C $\cup\ G(v_i, e_i, w_i)$;
11 **fin**
12 $L_c \leftarrow \varnothing$;
13 **para cada** $c_i \in C$ **hacer**
14 $l_i \leftarrow$ contar la cantidad de vértices en c_i;
15 $L_c \leftarrow L_c \cup l_i$
16 **fin**
17 $Q_{L_c} \leftarrow$ calcular los cuantiles de los l_i únicos $\in L_c$;
18 $C_{q_{alta}} \leftarrow$ seleccionar las c_i cuya $l_i \in Q_{alta}$.;
19 $C_{q_{critica}} \leftarrow$ seleccionar las c_i cuya $l_i \in Q_{alta}$.;
20 $C_s \leftarrow$ seleccionar las $c_i \in Q_{alta} | Q_{critica}$ con el mayor número de vértices $User$;
Output: C_s

4.4. Detección de comunidades significativas

Una comunidad es un subgrafo $C \in G$, conformado por nodos *User* y *Tweet* altamente interconectados. Sin embargo, no se requiere analizar todas las comunidades C para identificar lideres de opinión. Las comunidades son agrupan en comunidades significativas C_s y no significativas C_{ns}, de acuerdo al número de nodos *User*.

En el Algoritmo 3, se identifican todas las comunidades C existentes en el grafo $G(V, E, W)$ a partir del algoritmo *Componentes fuertemente conectados*, produciendo un grafo G_{cc} donde cada vértice es etiquetado por el identificador de la comunidad donde pertenece (líneas 2-3, Algoritmo 3). Luego, se extraen todos los vértices etiquetados V_L y se crea una lista única de identificadores de comunidades $list\_cc$. A partir de ella, se filtran los nodos o vértices *User* y *Tweet* para crear el conjunto de comunidades C (líneas 4-11, Algoritmo 3).

Con el conjunto de comunidades C se crea la lista L_c contabilizando el número de nodos n_i existente en cada comunidad c_i. La lista L_c almacena los tamaños (cantidad de

nodos) de cada comunidad, eliminándose los valores repetidos (líneas 12-16, Algoritmo 3). Se realiza un análisis de cuartiles a lista L_c para segmentar las comunidades en cuatro regiones de prioridad de análisis: baja, moderada, alta y crítica. Las comunidades c_i cuyo tamaño l_i pertenezca a la región Q_{alta}, región de prioridad alta, y a $Q_{critica}$, región de prioridad crítica; son elegidas para seleccionar las comunidades significativas de acuerdo al número de nodos *User* presente en cada comunidad (líneas 17-20, Algoritmo 3). Como resultado del algoritmo, se obtiene el conjunto de comunidades significativas C_s del grafo $G(V, E, W)$.

Para la implementación del algoritmo de detección de comunidades se utilizó el lenguaje de programación funcional Scala y el algoritmo *Connected components* implementado en GraphX. Los datos de los nodos y enlaces de cada comunidades significativas $c_{s_i} \in C_s$ serán almacenadas en el directorio HDFS */graph/community_cs/cs_id/t_vertex/* y */graph/community_cs/cs_id/t_edge/*, respectivamente; donde *cs_id* es el identificador de comunidad significativa, y estarán disponibles para la identificación de los líderes de opinión explicada en la siguiente sección.

4.5. Identificación de líderes de opinión

Inspirado en el modelo de procedencia de datos (PROV-DM, por sus siglas en ingles), se propone un modelo de interacción para cuantificar la influencia global ejercida en una red social. En este modelo, la influencia es ejercida de forma directa e indirecta a través de la difusión de tweets, como se muestra en la Figura 4.4.

Con respecto al esquema, existen tres usuarios U_i, U_j y U_k relacionados por la interacción de sus tweets T_i, T_j y T_k. El proceso de interacción se da de la siguiente manera: Si el contenido del tweet T_i, cuyo propietario es el usuario U_i, es de interés para el usuario U_j y provoca la creación del tweet T_j por una acción de *retweet* o *quoted*, el usuario U_i transfiere parte de su influencia al usuario U_j. Si el usuario U_k genera el tweet T_k en base la contenido difundido por el tweet T_j, el usuario U_i influye en el U_k a través del usuario U_j. Por lo tanto, el usuario U_i influye directamente en el usuario U_j e indirectamente en el usuario U_k.

La influencia que puede ejerce un usuario en una red social por interacción estará conformada por la transferencia directa (ID) o indirecta (II) de su influencia. Además, otro factor a considera es la reputación del usuario, la cuál incentiva la transferencia de influencia mas no es un factor determinante.

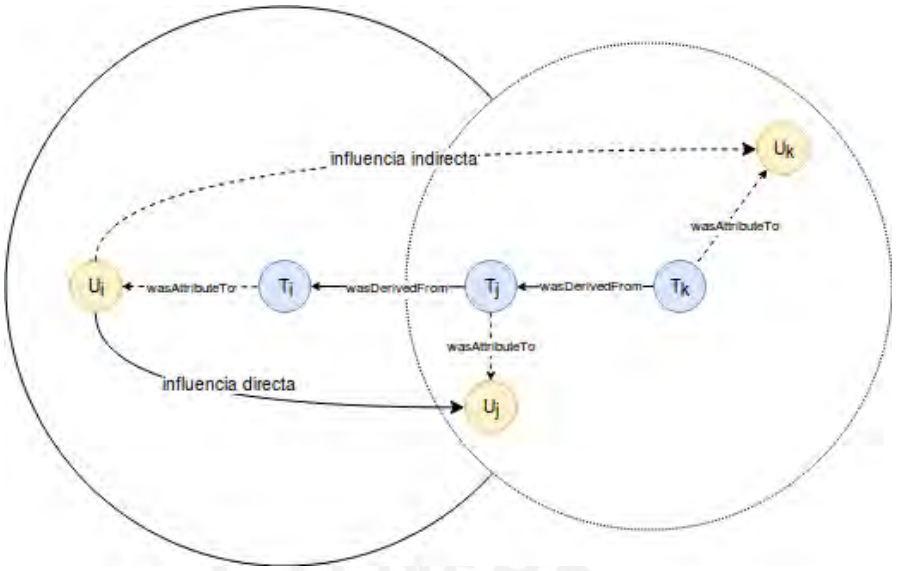


FIGURA 4.4: Esquema de interacción del usuario en la red social.
Fuente: Elaboración propia.

De la Figura 4.4, la influencia directa que un usuario i ejerce en una comunidad c_s se define como:

$$ID_{ic_s} = \frac{\sum_{j=1}^{N_j^{r=1}} w_{ij}}{\text{máx}\{\sum_{v \in N_u^{r=1}} w_{uv}\}} \tag{4.2}$$

Donde, $N_{r=1}$ es el número total de usuarios j que interactúan con el usuario i por medio de tweets que tienen una distancia r igual a 1, y w_{ij} es el peso del enlace $WasDerivedFrom$ entre los nodos tweets. El denominador de la Ecuación 4.2 es la máxima suma de pesos w_{uv} entre los nodos tweet de un usuario v en reacción a los tweet del usuario u con un distancia r de 1.

Cuando la distancia entre los tweets entre dos usuarios es mayor a 1, se calcula la influencia indirecta transmitida entre los nodos tweets. La influencia indirecta que ejerce un usuario i en una comunidad c_s está definida en la Ecuación 4.3.

$$II_{ic_s} = \frac{\sum_{j=1}^{N_j^{r>1}} w_{ij}}{\text{máx}\{\sum_{v \in N_u^{r>1}} w_{uv}\}} \tag{4.3}$$

Donde, $N_{r>1}$ es el número total de usuarios j que interactúan con el usuario i por medio de tweets que tienen una distancia r mayor a 1, y w_{ij} es el peso del enlace $WasDerivedFrom$ entre los nodos tweets. Por lo tanto, una mayor distancia entre los nodos usuarios conlleva una mayor poder de influencia para el usuario i.

Así, la influencia global que ejerce un usuario i dentro de una comunidad c_s está definida en la Ecuación 4.4.

$$I_i = \alpha ID_i + (1 - \alpha) II_i \tag{4.4}$$

Donde se utiliza el atributo estático *text* del tweet para generar el peso w entre los tweets y los atributos de interacción del usuario generadas en el proceso de construcción de la red social. El parámetro α es un valor entre 0 y 1 que indica la relevancia dada a los dos tipos de influencia estudiadas.

Para la implementación del algoritmo de identificación de líderes de opinión se utilizó el lenguaje Scala en su versión 2.11 en conjunto con GraphX. El proceso algoritmo para calcular la influencia de los usuarios se muestra en Algoritmo 4. En resumen, para obtener a los n lideres de opinión de cada comunidad c_p identificada en la red social, se utilizan la Ecuación 4.4 para cuantificar la influencia global de los usuarios.

Algoritmo 4: Selección de Lideres de Opinión.

Input : C_s: conjunto de comunidades significativas.;
α: número especificado por el usuario.;
k: número de líderes de opinión por comunidad significativa c_s.
Output: OL_{cs}: conjunto de líderes de opinión en cada comunidad significativas c_s.

```
1  OL_cs ← ∅;
2  para cada c_s ∈ C_s hacer
3      L_cand ← ∅;
4      para cada u_i ∈ c_s hacer
5          di_u_i ← calcular influencia directa con la Ecuación 4.2;
6          ii_u_i ← calcular la influencia indirecta con la Ecuación 4.3;
7          ig_u_i ← calcular la influencia global con la Ecuación 4.4 y el parámetro α;
8          L_candc_s ← L_candc_s ∪ (u_i, ig_u_i)
9      fin
10     L_OLc_s ← extraer los primeros k usuarios con mayor ig_u de L_candc_s;
11     OL_cs ← OL_cs ∪ L_OLc_s;
12 fin
```

Output: OL_{cs}

Para mostrar los resultados obtenidos de la ejecución de los algoritmos definidos, se construye un componente visual, cuya implementación se detalla en la siguiente sección.

4.6. Visualización de resultados

Como se explicó en la Sección 2.3.6, existen varias librerías JavaScript para visualizar datos mediante grafos. Sin embargo, no todas las librerías se encuentran optimizadas para visualizar grandes volúmenes de nodos y enlaces. De acuerdo a Andrei Kashcha [65], la librería VivaGraphJS [45] presenta el mejor rendimiento para visualizar más de 1 000 nodos, con sus respectivos enlaces. Por lo tanto, para mostrar nuestros resultados, utilizamos la librería VivaGraphJS[3] dentro del entorno de ejecución proporcionado por Node.js[4] en su versión 6.12.2, debido a la facilidad de desarrollar componentes JavaScript ejecutables.

En la Figura 4.5 se muestra los pasos a seguir para visualizar los resultados obtenidos en el estudio de la identificación de líderes de opinión. Para renderizar la red social, VivaGraphJS leerá un archivo *graph_twitter.json* con los datos de los nodos *User* y *Tweet*, y los enlaces *WasAttributeTo* y *WasDerivedFrom*. El archivo *graph_twitter.json* se genera por la clase *ExportRddProcess.scala*, implementado en Scala, para transformar los resultados del Algoritmo 4 al formato JSON esperado, ver un ejemplo del formato en el Apéndice B. Así, a través del programa *GraphVisual*, que implementamos utilizando Node.js, se representará nuestra red social estudiada como un grafo, donde los nodos amarillos representan a los usuarios y los nodos verdosos, a los tweets.

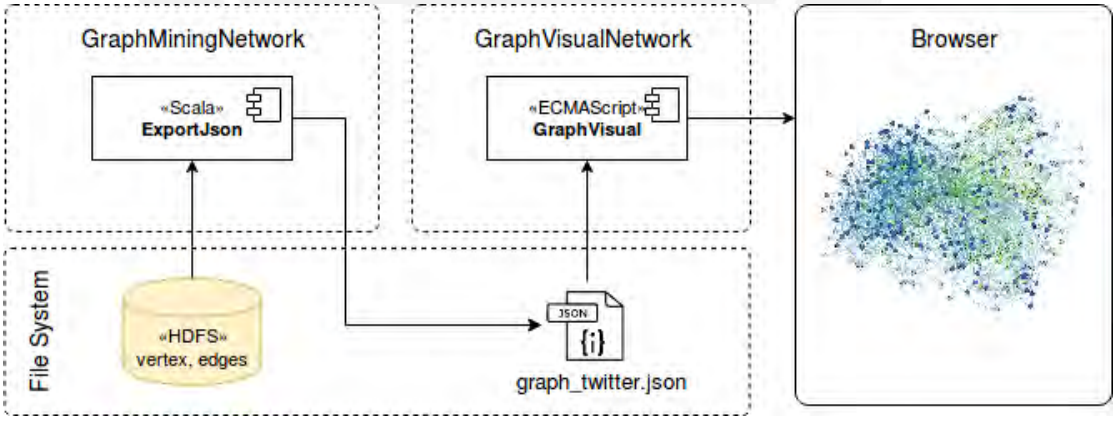


FIGURA 4.5: Proceso de visualización de grafos.
Fuente: Elaboración propia.

En el experimento se utilizará el presente proceso para mostrar los resultados obtenidos de la ejecución de los algoritmos de construcción de red social, la detección de comunidades y el cálculo de influencia de los usuarios. En la Figura 4.6 se puede apreciar el resultado de la ejecución del algoritmo de construcción de red social, a través de este

[3]VivaGraphJS: https://github.com/anvaka/VivaGraphJS
[4]NodeJS: https://nodejs.org/es/

proceso. La red social tiene un total de 26 184 nodos (13 897 nodos *Tweet* y 12 287 *User*) y 26 920 enlaces (13 897 enlaces *WasAttributeTo* y 13 023 enlaces *WasDerivedFrom*).

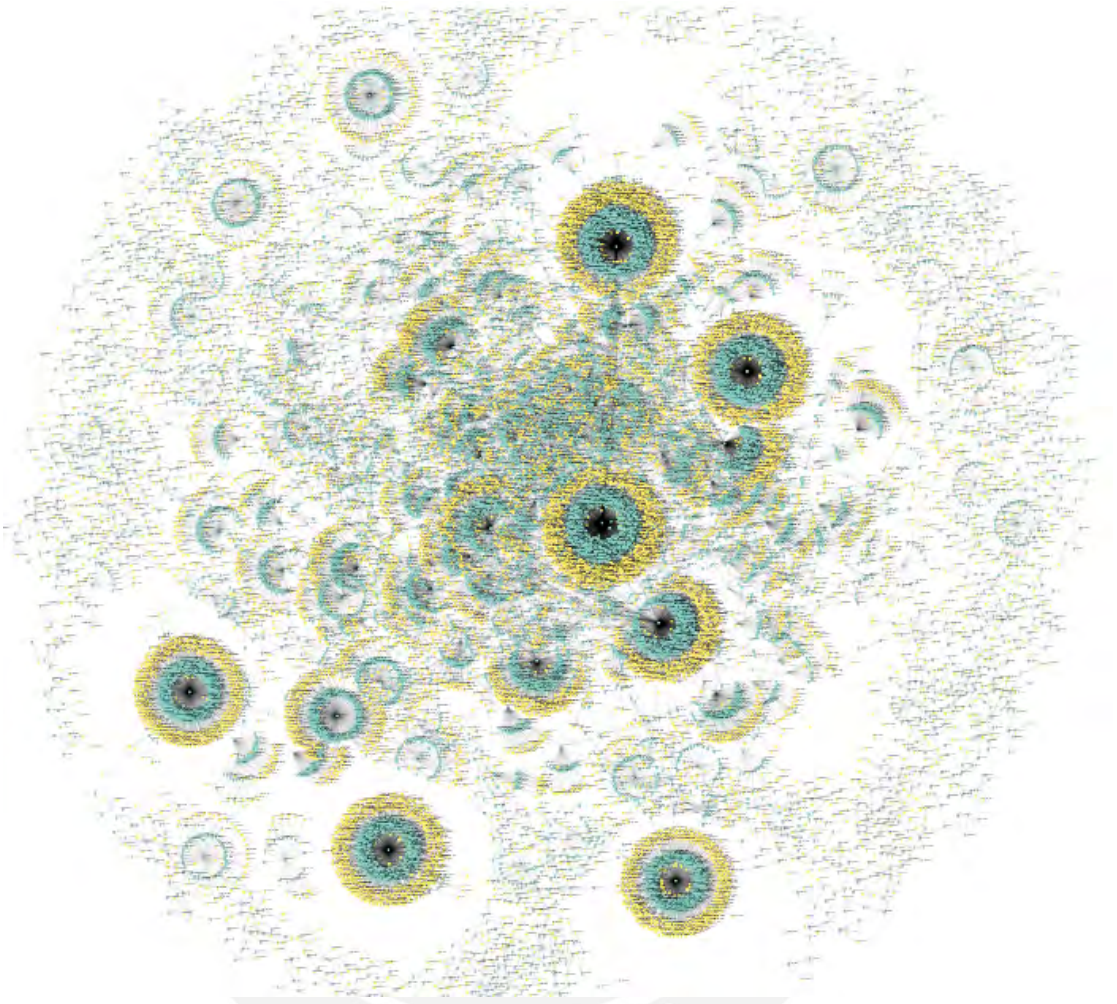


FIGURA 4.6: Representación gráfica de red social mediante un grafo dirigido de 26 184 nodos y 26 920 enlaces.
Fuente: Elaboración propia.

En resumen, en el presente capítulo se describieron los criterios y algoritmos utilizados para obtener los resultados esperados en el trabajo de investigación. Luego de implementar con los componentes y algoritmos mencionados, se procede a realizar la experimentación descrita en el Capítulo 5, el cuál explicaremos a continuación.

Capítulo 5

Experimentos y Resultados

El objetivo del presente capitulo es de aplicar a los datos extraídos de Twitter, la construcción del grafo que represente nuestra red social estudiada, la segmentación del grafo, la identificación de los lideres de opinión como se ejemplifica en la Figura 5.1.

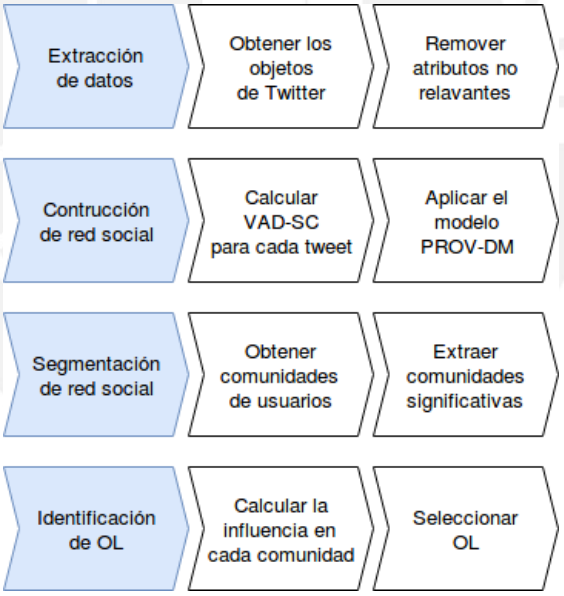


FIGURA 5.1: Diseño del experimento conformada por ocho tareas (casilla s de color blanco) agrupadas en cuatro fases(casillas de color blanco).
Fuente: Elaboración propia.

Los procedimientos se ejecutaron sobre una computadora portátil con procesador Intel Core i7 2.40 GHz, 16 GB de RAM y sistema operativo Ubuntu[1] 16.04 LTS. Para la visualización de grafos se utilizará el navegador web Mozilla Firefox Quantum 61.0.1.

[1]Ubuntu: https://www.ubuntu.com/

En las siguientes secciones se describe la ejecución de los procedimientos y algoritmos establecidos en cada fase del experimento diseñado en la Figura 5.1. En la sección 5.1 se describe la extracción los objetos *User* y *Tweet* a partir de los objetos JSON recolectados desde Twitter. En el sección 5.2; se describe la transformación de los objetos *User* y *Tweet* en nodos *User*, nodos *Tweet*, enlaces *WasAttributeTo* y enlaces *WasDerivedFrom*, para construir el grafo que representará la red social estudiada. En la sección 5.3 se describe la extracción de las comunidades significativas del grafo construido y, en la sección 5.4 se identifican a los líderes de opinión de cada comunidad significativa. Finalmente, en la sección 5.5 se compara los resultados del algoritmo propuesto con el algoritmo PageRank [57] en términos del número máximo de nodos influenciados por los líderes de opinión identificados en cada algoritmo.

5.1. Extracción de datos

El conjunto de datos utilizados en el experimento consta de 1 589 131 objetos JSON extraídos desde Twitter. Considerando solo los objetos JSON que tengan un tweet en inglés en su contenido, se crean 768 287 objetos *User* y 1 326 146 objetos *Tweet*. El objeto *Tweet* esta definido por sus atributos: *id*, *text*, *lang* y *created_at*; y el objeto *User* por sus atributos: *id* y *created_at*, como se observa en la Tabla 5.1.

Atributos	Tipo	Objeto
id	Long	User
created_at	String	User
id	Long	Tweet
text	String	Tweet
lang	String	Tweet
created_at	String	Tweet

Tabla 5.1: Atributos relevantes de los objetos *User* y *Tweet*.
Fuente: Elaboración propia

Luego, se realizó el análisis temporal del atributo *created_at* del objeto *Tweet* para definir si los datos obtenidos desde Twitter representan una muestra significativa del total de tweets relacionados al tema del calentamiento global generados en Twitter. El análisis demostró que la tasa de recolección de tweets por minuto no supera los 3 500 tweets por minuto, ver Figura 5.2. Por lo tanto, de acuerdo a Andy Piper [15], el conjunto de datos utilizados en el experimento representa el 100 % de datos generados por los usuarios que reaccionaron al tema *Calentamiento Global*, entre el 03/07/2018 y el 23/09/2018.

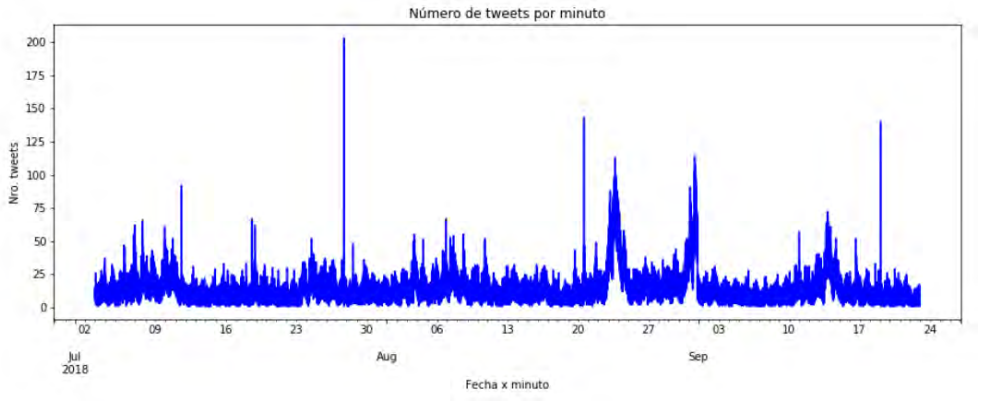


FIGURA 5.2: Número de tweets recolectados por minuto.
Fuente: Elaboración propia.

Definidos los atributos de los objetos *User* y *Tweet*, se inicia la fase de creación de la red social, explicada en la siguiente sección.

5.2. Construcción de red social

En esta fase se ejecutó el Algoritmo 2 para generar el grafo $G(V, E, V)$ que represente la red social de usuarios interesados en el calentamiento global. El Algoritmo 2 crea enlaces *WasAttributeTo* y *WasDerivedFrom* a partir de los objetos JSON procesados en la fase anterior. Al finalizar la ejecución del algoritmo se obtuvo lo siguiente:

- 768 287 nodos *User* y 1 326 146 nodos *Tweet*, $\in V$.
- 1 326 146 enlaces *WasAttributeTo* y 996 957 enlaces *WasDerivedFrom*, $\in E$.
- Enlaces con pesos $w \in W$, asignados de acuerdo a la Ecuación 4.1.

Para mostrar las interacciones creadas a partir del Algoritmo 2 se ejecutó el programa *GraphVisual*. Sin embargo, debido a recursos computacionales limitados, se creó un grafo G_{o100} con 100 000 objetos JSON. El grafo obtenido se muestra en la Figura 5.3. El grafo G_{o100} esta conformado por 83 372 nodos *User* y 103 396 nodos *Tweet*. Además, entre los nodos *User* y *Tweet* se creó 103 396 enlaces *WasAttributeTo* y 74 084 enlaces *WasDerivedFrom* entre los nodos *Tweet*. Además, los nodos *User* son representados con color amarillo y los nodos *Tweet*, con color verdoso. Como se aprecia en la Figura 5.3, en el grafo G_{o100} existen grupos de nodos conformados por nodos *User* y *Tweet* que tienen un mayor tamaño (número de nodos) en comparación a otros grupos de nodos posicionados alrededor, formando una corona.

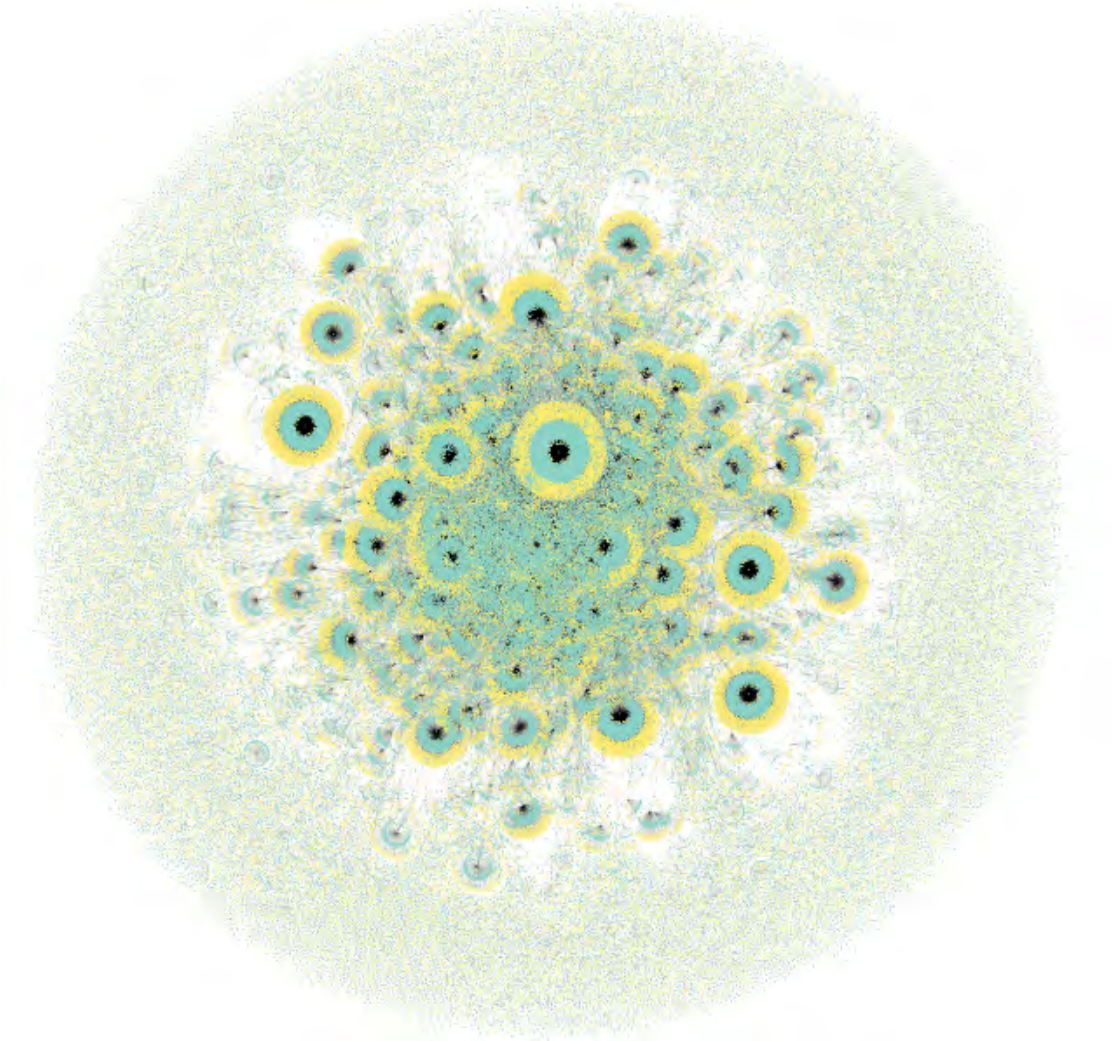


FIGURA 5.3: Grafo G_{o100} con 57,748 nodos y 53,150 enlaces.
Fuente: Elaboración propia.

Luego de obtener el grafo original G, se procedió a segmentar el grafo en comunidades significativas para reducir el costo de computo en cuantificar la influencia de los nodos *User*, como se explica a continuación.

5.3. Segmentación de red social

Para esta fase se utilizó el grafo $G(V, E, W)$ obtenido del proceso de construcción de la red social, obteniendo comunidades ó módulos significativos dentro del grafo. Aplicamos los pasos del Algoritmo 3, identificando 116 808 comunidades (o grupos) de nodos *User* y *Tweet*. Luego, identificadas las comunidades significativas, se aplicó el análisis de cuartiles con el fin de asignar prioridades a las comunidades identificadas según su

concentración (número) de nodos. Aplicando la función cuartil a la lista ordenada de

Clasificación de prioridades de tamaño de comunidades c_i (Análisis de cuartiles)																				
Tamaños	**Prioridad baja**					**Prioridad moderada**					**Prioridad alta**					**Prioridad crítica**				
	g_1	g_2	...	g_{26}	g_{27}	g_{28}	g_{29}	...	g_{52}	g_{53}	g_{54}	g_{55}	...	g_{79}	g_{80}	g_{81}	g_{82}	...	g_{106}	g_{107}
	2	3	...	27	28	29	30	...	54	56	57	58	...	104	105	110	115	...	1096	1768477
				1.° Cuartil = 28.5					2.° Cuartil = 57.0					3.° Cuartil = 107.5						

TABLA 5.2: Clasificación de prioridades de tamaño de comunidades c_i.
Fuente: Elaboración propia

tamaños de las comunidades L_{c_i}, se obtuvieron tres cuartiles: $Q_1 = 28.5$, $Q_2 = 57.0$ y $Q_3 = 107.5$, ver Tabla 5.2. A partir de ello, se distribuyó las comunidades en cuatro prioridades, de acuerdo a la concentración de nodos usuarios. Las comunidades son etiquetas en comunidades de prioridad baja, moderada, alta y crítica. Para nuestro trabajo, se seleccionó las comunidades de prioridad alta y crítica debido a la mayor probabilidad de encontrar líderes de opinión dentro de grupos con mayor concentración de nodos usuarios.

Seleccionado los grupos, se analizó la concentración de nodos usuarios de las comunidades de prioridad crítica, evidenciándose que la comunidad c_1 es la comunidad más significativa del grafo G, ver Figura 5.4. Como la concentración de usuarios de la comunidad c_1 es mucho mayor que las presentadas en otras comunidades, se realizó un segundo análisis sin considerar a la comunidad c_1. Los resultados obtenidos se muestran en la Figura 5.5, donde se muestra el análisis de frecuencia de usuarios en las comunidades de prioridad críticas (barras de color rojo en la Figura 5.5) y las comunidades de prioridad alta (barras de color azul en la Figura 5.5). Como resultado del segundo análisis, se identificó como comunidades significativas a las comunidades: c_6, c_{11}, c_{20}, c_{26}, c_{27}, c_{30}, c_{35}, c_{43} y c_{45}.

Finalmente, juntando el primer y segundo análisis realizado, el conjunto de comunidades significativas del grafo G se define como:

$$C_s = \{c_1, c_6, c_{11}, c_{20}, c_{26}, c_{27}, c_{30}, c_{35}, c_{43}, c_{45}\} \tag{5.1}$$

Además, se identificó que la comunidad Cs_1 tiene el mayor número de nodos y enlaces; 630 365 y 1 138 112 respectivamente. Lo sigue la comunidad c_6 conformada por 145 nodos *Usuario*, 163 nodos *Tweet*, 163 enlaces *WasAtttributeTo* y 160 enlaces *WasDerivedFrom*; la comunidad c_{11} con 89 nodos *Usuario*, 116 nodos *Tweet*, 116 enlaces *WasAtttributeTo* y 108 enlaces *WasDerivedFrom*; la comunidad c_{20} con 74 nodos *Usuario*, 77 nodos *Tweet*, 77 enlaces *WasAtttributeTo* y 73 enlaces *WasDerivedFrom*; la comunidad c_{26} con

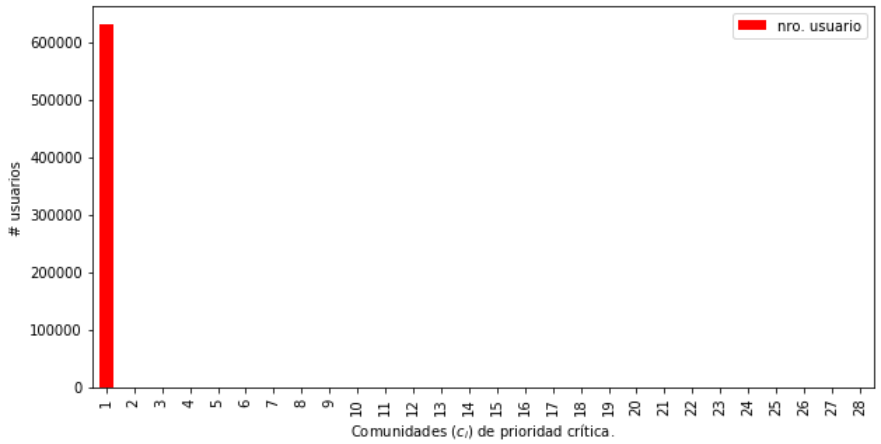


FIGURA 5.4: Análisis de frecuencia de usuarios en las comunidades de prioridad crítica (color rojo).
Fuente: Elaboración propia.

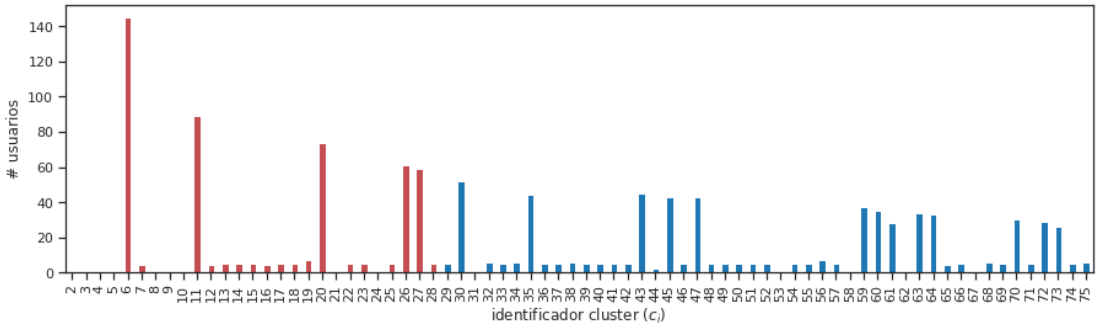


FIGURA 5.5: Análisis de frecuencia de usuarios en las comunidades de prioridad crítica (color rojo) y de prioridad alta (color azul).
Fuente: Elaboración propia.

61 nodos *Usuario*, 63 nodos *Tweet*, 63 enlaces *WasAtttributeTo* y 60 enlaces *WasDerivedFrom*; así sucesivamente como se muestra en la Tabla 5.3.

Cada comunidad significativa C_si es representada por un grafo G_{C_si} a partir de sus nodos y enlaces almacenados en los RDD's. Los datos de cada comunidad significativa C_si se almacenan en rutas HDFS distintas. Por ejemplo: la ruta HDFS asignada para la comunidad cs_1 es $/graph/community/cs1/$; por lo tanto, los datos de los nodos se guardan en la ruta $/graph/community/cs1/t\_vertex$ y los datos de los enlaces, en la ruta $/graph/community/cs1/t\_edge$.

Para visualizar las comunidades significativas C_si, nuestro componente de visualización apunta a las rutas HDFS de cada comunidad. En la Figura 5.6 se muestran las representaciones gráficas de 9 de las 10 comunidades significativas de la Tabla 5.3 que se

C_s	c_i	nro. nodos Usuario	nro. nodos Tweet	**Total nodos**	nro. enlaces WasAttributeTo	nro enlaces WasDerivedFrom	**Total enlaces**
1	1	630 365	1 138 112	1 768 447	1 138 112	996 889	2 135 001
2	6	145	163	308	163	160	323
3	11	89	116	205	116	108	224
4	20	74	77	151	77	73	150
5	26	61	63	124	63	60	123
6	27	59	59	118	59	56	115
7	30	52	55	107	55	52	107
8	35	44	63	107	63	49	112
9	43	45	45	90	45	44	89
10	45	43	46	89	46	45	91

TABLA 5.3: Métricas de nodos y enlaces en las 10 comunidades más significativas. Fuente: Elaboración propia.

obtuvieron al aplicar nuestro componente visual. Como se mencionó en secciones anteriores, los usuarios son representados con nodos de color amarillo y los tweets, con nodos de color verdoso.

Debido a que aplicamos el procesamiento gráfico de los grafos en el navegador web Mozilla Firefox[2] y Google Chrome[3], el número de nodos y enlaces procesados esta limitado a aproximadamente 250 000 nodos y 300 000 enlaces. Por lo tanto, en la Figura 5.6 no se incluye la visualización de la comunidad significativa C_s1.

La identificación de comunidades significativas C_s da inicio al cálculo de la influencia ejercida por cada nodo *User*, cuya ejecución se explica en la siguiente sección.

5.4. Identificación de líderes de opinión

Cada comunidad significativa obtenida en el proceso anterior sirve de entrada al procedimiento de identificación de líderes de opinión para cuantificar la influencia de cada usuario en su determinada comunidad. La ejecución del Algoritmo 4 retornó un ranking de los usuarios más influyentes en base a su influencia directa e indirecta. En la ejecución del Algoritmo 4 se utilizó un $\alpha = 0.5$ para balancear la influencia directa e indirecta. Los resultados obtenidos se muestran en la Tabla 5.4.

En caso que un nodo *User* influyera a otros nodos usuarios utilizando nodos *Tweet* diferentes, se considera como influencia directa o indirecta, el tweet con la mayor influencia acumulada. Por ejemplo, en la comunidad significativa C_s45, el nodo de usuario identificado con el código -855007560151068198 utiliza 5 nodos tweets diferentes para influenciar a otros nodos usuarios. En la Tabla 5.4 se muestra la lista de los usuarios

[2]Mozilla Firefox: https://www.mozilla.org/es-ES/firefox/
[3]Google Chrome: https://www.google.com/chrome/

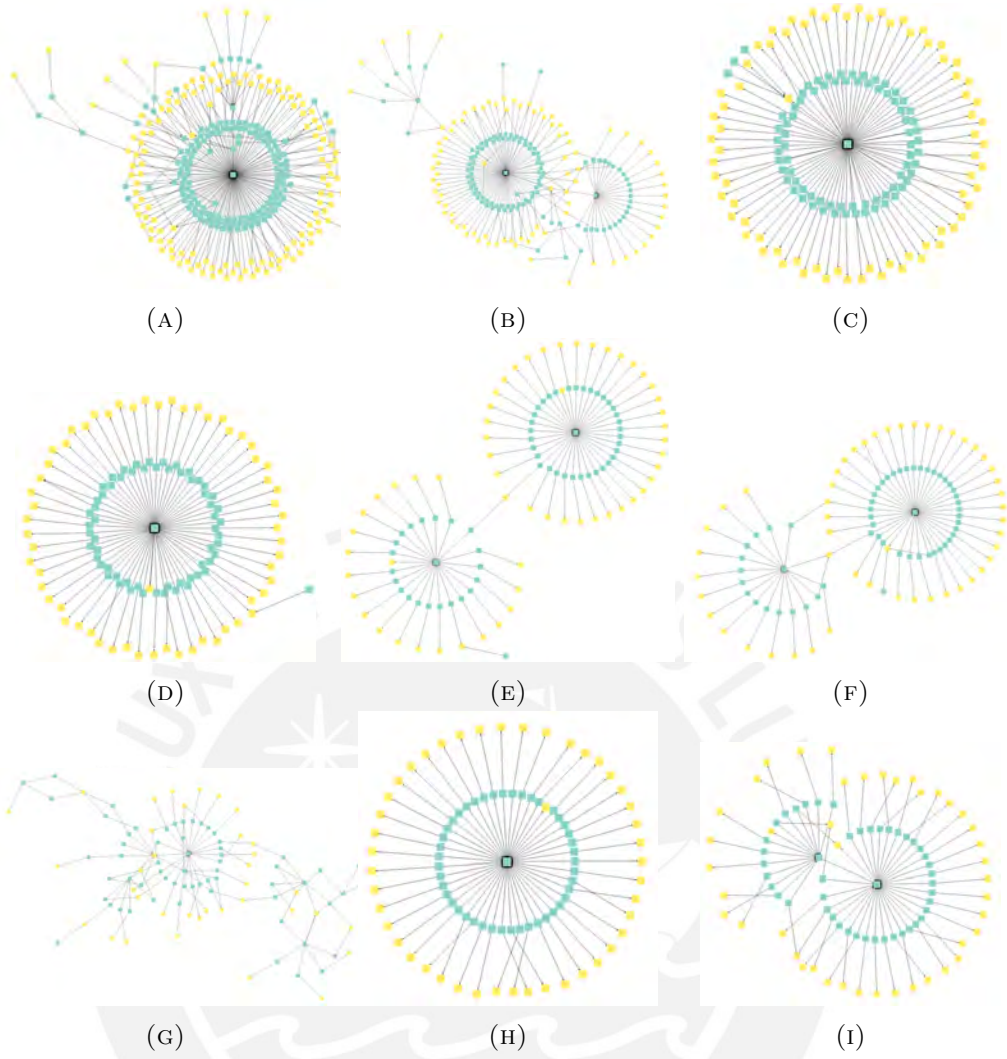


FIGURA 5.6: La representación gráfica de las comunidades significativas: G_{C_s6} (A), G_{C_s11} (B), G_{C_s20} (C), G_{C_s26} (D), G_{C_s27} (E), G_{C_s30} (F), G_{C_s35} (G), G_{C_s43} (H) y G_{C_s45} (I).
Fuente: Elaboración propia.

más influyentes obtenidos en cada comunidad significativa C_si y están sombreados de color gris. Además, se identificó que las comunidades significativas C_s20, C_s26 y C_s43 solo tienen un líder de opinión.

De acuerdo a los resultados obtenidos, observamos que el algoritmo propuesto mostró la participación oculta de nodos usuarios no considerados en estudios centrados en calcular la influencia directa. Por ejemplo, en la comunidad significativa C_s35, el usuario identificado con el código 7993878106633819059 se convierte en líder de opinión si solo se considera su influencia directa. En cambio, si adicionalmente consideramos la influencia indirecta, el usuario -8550075601510681981 se convierte en el usuario de mayor influencia dentro de la comunidad significativa C_s35. Este mismo cambio se aprecia en la comunidad significativa C_s45 entre los usuarios 9140203086483427515 y -6301844523572966283.

C_s	**Identificador nodo User**	**Influencia Directa**	**Influencia Indirecta**	**Influencia Total**
6	1793992859971553135	1	1	1
6	8173025407379598472	0.08	0.46	0.27
6	3716013776701647659	0.07	0.45	0.26
6	6128976145273286571	0.06	0.45	0.255
6	-7399896575803150135	0.05	0.45	0.25
6	7362866210696515113	0.05	0.45	0.25
6	6707547958452661761	0.02	0.45	0.235
6	-1491771540576983153	0.02	0.45	0.235
6	7806644684165941728	0.01	0.45	0.23
11	-6960804210000046898	1	1	1
11	3596611392670047601	0.1	1	0.55
20	-7012288462872525403	1	1	1
26	-5568738118831847322	1	1	1
27	6891054584377253686	1	1	1
27	6066187218255139633	0.57	1	0.785
30	-898340183019586312	1	0.88	0.94
30	-6539193335341611866	0.46	1	0.73
35	-8550075601510681981	0.7	1	0.85
35	7993878106633819059	1	0.53	0.765
35	3157963772436753634	0.09	0.69	0.39
35	-8584689512450977976	0.08	0.56	0.32
35	-7937633325400586379	0.04	0.31	0.175
35	-2446178784232109486	0.04	0.27	0.155
43	-9061945922882526848	1	1	1
45	6431509402151977296	1	0.44	0.72
45	9140203086483427515	0.05	1	0.525
45	-6301844523572966283	0.38	0.32	0.35

TABLA 5.4: Lista de los usuarios más influyentes en 9 de las 10 comunidades más significativas.
Fuente: Elaboración propia.

Para interpretar el resultado obtenido, se visualiza el comportamientos de los usuarios de la comunidad significativa C_s35 en la Figura 5.7. En la Figura, se logra apreciar que la influencia del usuario -8550075601510681981 se propaga a través de la creación de solo 3 tweets (nodos de color verdoso). Además, los usuarios influenciados comparten la idea u opinión expresada por el usuario -8550075601510681981 porque presentaron un peso $w_{ij} = 1{,}9$ entre sus nodos tweets, significando una influencia positiva sobre el tema del calentamiento global. Por otro lado, los mensajes compartidos por el usuario 7993878106633819059 influyen negativamente en los usuarios de la comunidad, $w_{ij} = 0{,}74$.

La consideración de la influencia indirecta en la determinación de los usuarios más influyentes modifica en un 33 % las posiciones de los líderes de opinión, con un parámetro

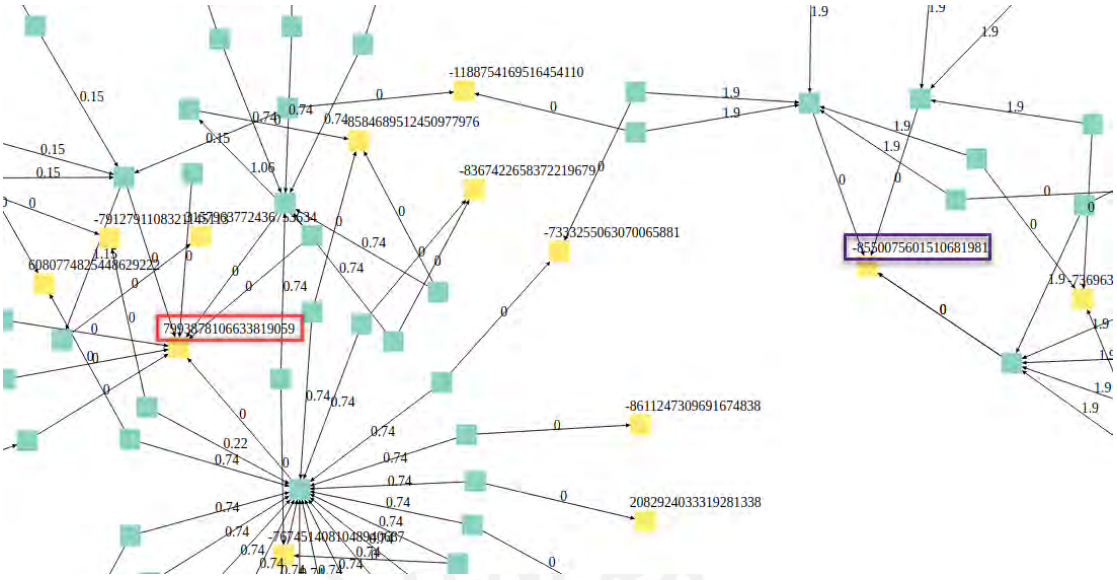


FIGURA 5.7: Representación gráfica de la influencia ejercida por el usuario -8550075601510681981 (enmarcado de color morado) y el usuario 7993878106633819059 (enmarcado de color naranja) en la comunidad significativa C_s35.

α de 0.5. Sin embargo, el ratio puede ser mayor si en nuestro análisis priorizamos la influencia indirecta sobre la influencia directa, asignando valores α mayores a 0.5.

5.5. Evaluación de resultados

En esta sección se compara los resultados obtenidos con otras investigaciones relacionadas. Como se observo en el Estado del Arte, Capítulo 3, los algoritmos revisados están fuertemente relacionados a los atributos utilizados para cuantificar la influencia, y en otros casos, por la estructura de datos utilizada para representar la red social estudiada. Por lo tanto, para comparar nuestro trabajo utilizaremos la métrica de difusión de influencia, la cual se decide el mejor algoritmo en base a la mayor cantidad de nodos que puede influenciar los líderes de opinión obtenido por cada algoritmo.

Segmento	**Nro. de comunidades** c_i	
	G_{UT}	G_U
Prioridad baja	116 611	1 051 309
Prioridad moderada	125	447
Prioridad alta	46	210
Prioridad crítica	29	159

TABLA 5.5: Comparación del número de comunidades c_i calculado en el grafo propuesto G_{UT} y el grafo PageRank G_U.
Fuente: Elaboración propia.

Como en otros trabajos de investigación [58] y [25], compararemos nuestros resultados en relación al algoritmo PageRank [57]. Sin embargo debido a que el algoritmo PageRank se aplica en grafos homogéneos, es decir en grafos con un solo tipo de nodo, aplicamos nuestro proceso de análisis en un grafo homogéneo G_U generado solamente con datos de los usuarios.

En el nuevo proceso de análisis, en el grafo homogéneo G_U se obtuvo mayor número de comunidades en cada uno de los segmentos estudiados del grafo heterogéneo G_{UT}, ver Tabla 5.5. El incremento de comunidades se debe a que el grafo G_U solo utiliza datos propios de los usuarios para estudiar su influencia, ignorando la participación de los contenidos (tweets) generados en la difusión de influencia.

$C_s i$	**Líderes de opinión**	**PageRank**	**Propuesta**
6	1793992859971553135	129	134
6	8173025407379598472	7	7
11	-6960804210000046898	1	1
11	3596611392670047601	102	102
20	-7012288462872525403	73	73
26	-5568738118831847322	60	60
27	6891054584377253686	35	35
27	6066187218255139633	21	21
30	-898340183019586312	37	37
30	-6539193335341611866	15	15
35	-8550075601510681981	13	14
35	7993878106633819059	31	31
43	-9061945922882526848	44	44
45	6431509402151977296	31	44
45	9140203086483427515	1	45

TABLA 5.6: Comparación de número de nodos influenciados por los líderes de opinión, obtenidos al aplicar el algoritmo PageRank y el algoritmo propuesto.
Fuente: Elaboración propia.

Para contrastar los cambios que genera la inclusión del estudio de la procedencia de influencia, se comparó el número de nodos usuarios que los líderes de opinión, identificados por los algoritmos evaluados, logran influenciar en cada comunidad significativa $C_s i$. Tomando como referencia a los nodos usuarios etiquetados como líderes de opinión por nuestro algoritmo propuesto, aplicamos el algoritmo PageRank a estos nodos usuarios y con los resultados obtenidos, comparamos el número de nodos usuario influenciados en los dos enfoques en la Tabla 5.6. Como se evidencia en la tabla, la consideración de la procedencia de influencia permitió identificar un mayor número de usuarios influenciados y la reacción generada por cada tweet compartido. Por ejemplo, con el algoritmo PageRank la cuantificación de la influencia del usuario 9140203086483427515 esta basada en los datos generados por la interacción con un único usuario en cambio, con el

enfoque propuesto, se cuantifica en base a todos los usuarios que reaccionan a contenido expresado en los tweets, sea de forma positiva o negativa.

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo indicamos las conclusiones del trabajo realizados, los casos de uso donde se puede aplicar nuestro algoritmo y los siguientes trabajos futuros para mejorar los resultados obtenidos.

6.1. Conclusiones

En el trabajo se presenta cuatro contribuciones. Primero, la definición de un proceso de minería de grafos grandes (Big Graph Mining, por su traducción al inglés) para estudiar la influencia en grandes volúmenes de nodos y vértices. Segundo, la integración del modelo de procedencia de información PROV-DM en el estudio de la influencia social, permitiendo representar la interacción de los usuarios de Twitter en un grafo heterogéneo compuesto por dos tipos de nodos: *User* y *Tweet*. Este nuevo enfoque nos permitió estudiar a los tweets como objetos conductores de influencia hacia otros usuarios, además en la generación del grafo, permite representar la traza de como se propaga la influencia de cada usuario dentro de su comunidad, proporcionando mayor información para cuantificar su influencia total. Tercero, el desarrollo de un algoritmo para cuantificar la influencia de los usuarios, dentro de un gran grafo heterogéneo, en base a la influencia directa e indirecta presente en sus trazas de difusión de influencia. Por último, el desarrollo de un componente software para realizar análisis visual con el fin de interpretar los resultados obtenidos por nuestro algoritmo; empleando para ello la librería VivaGraphJS y el navegador web Google Chrome.

Por otro lado, la comparación del algoritmo propuesto con el algoritmo PageRank demostró que la información proporcionada por la influencia indirecta no solo permite cuantificar con mayor precisión a los líderes de opinión, sino también permite identificar

grupos de usuarios con la capacidad de maximizar la influencia positiva o negativa dentro de las comunidades sociales virtuales. Este grupo de usuarios pueden emplearse para propagar información de manera oportuna, organizada y eficiente entre las comunidades de usuarios conectados a un medio social, en actividades de preparación, reacción y recuperación frente a una situación de emergencia y desastre.

6.2. Trabajos Futuros

Se sugiere identificar más entidades involucradas en la propagación de la influencia dentro de las comunidades de usuarios, para utilizar el modelo PROV-DM completo en la representación de la red social más cercana a la realidad. Además, replicar el experimento utilizando datos de otras plataformas de redes sociales como Facebook, Instagram[1] y Pinterest para estudiar la transcendencia de la influencia de los líderes de opinión entre dos o más redes sociales en un contexto homogéneo (tópicos relacionados).

Además, se sugiere mejorar el componente visual utilizando motores gráficos dedicados como Unity o Unreal Engine para obtener grafos grandes que permitan mostrar el comportamiento de los líderes de opinión entre dos o más redes sociales.

[1]Instagram: https://www.instagram.com

Bibliografía

[1] Yuwei Jia, Kun Chao, Xinzhou Cheng, Mingqiang Yuan, and Mingjun Mu. Big data based user clustering and influence power ranking. In *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*, pages 371–375, Sept 2016. ISBN 978-1-5090-4100-8. doi: 10.1109/ISCIT.2016.7751653.

[2] Min Song, Meen Chul Kim, and Yoo Kyung Jeong. Analyzing the political landscape of 2012 korean presidential election in twitter. *IEEE Intelligent Systems*, 29(2):18–26, Mar 2014. ISSN 1541-1672. doi: 10.1109/MIS.2014.20.

[3] Muhammad Mazhar Ullah Rathore, Malik Junaid Jami Gul, Anand Paul, Ashraf Ali Khan, Raja Wasim Ahmad, Joel Rodrigues, and Spiros Bakiras. Multilevel graph-based decision making in big scholarly data: An approach to identify expert reviewer, finding quality impact factor, ranking journals and researchers. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2018. ISSN 2168-6750. doi: 10.1109/TETC.2018.2869458.

[4] Paul F. Lazarsfeld, Berelson Bernard, and Gaudet Hazel. The people's choice. how the voter makes up his mind in a presidential campaign. 1965. ISSN 0002-7162. doi: 10.1177/000271624926100137.

[5] Alexandra L'Heureux, Katarina Grolinger, Hany F. Elyamany, and Miriam A. M. Capretz. Machine learning with big data: Challenges and approaches. 5:7776–7797, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2696365.

[6] Gema Bello-Orgaz, Jason J. Jung, and David Camacho. Social big data: Recent achievements and new challenges. *Information Fusion*, 28:45 – 59, 2016. ISSN 1566-2535. doi: 10.1016/j.inffus.2015.08.005.

[7] Jun Zhou, Guiping Wu, Manshu Tu, Bing Wang, Yan Zhang, and Yonghong Yan. Predicting user influence under the environment of big data. In *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 133–138, April 2017. ISBN 978-1-5090-4498-6. doi: 10.1109/ICCCBDA.2017.7951898.

[8] Sancheng Peng, Guojun Wang, Yongmei Zhou, Cong Wan, Cong Wang, and Shui Yu. An immunization framework for social networks through big data based influence modeling. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2017. ISSN 1545-5971. doi: 10.1109/TDSC.2017.2731844.

[9] Sancheng Peng, Guojun Wang, and Dongqing Xie. Social influence analysis in social networking big data: Opportunities and challenges. *IEEE Network*, 31(1):11–17, January 2017. ISSN 0890-8044. doi: 10.1109/MNET.2016.1500104NM.

[10] Jianwu Wang, Daniel Crawl, Shweta Purawat, Mai Nguyen, and Ilkay Altintas. Big data provenance: Challenges, state of the art and opportunities. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2509–2516, Oct 2015. ISBN 978-1-4799-9926-2. doi: 10.1109/BigData.2015.7364047.

[11] Hootsuite. Digital in 2018, 2018. URL https://www.slideshare.net/wearesocial/digital-in-2018-global-overview-86860338. [Online; accessed 27-Nov-2018].

[12] Marco Avvenuti, Stefano Cresci, Fabio Del Vigna, and Maurizio Tesconi. Impromptu crisis mapping to prioritize emergency response. *Computer*, 49(5):28–37, May 2016. ISSN 0018-9162. doi: 10.1109/MC.2016.134.

[13] Huiji Gao, Geoffrey Barbier, and Rebecca Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, 26(3):10–14, May 2011. ISSN 1541-1672. doi: 10.1109/MIS.2011.52.

[14] Twitter Developer. Filter realtime tweets, 2017. URL https://developer.twitter.com/en/docs/tweets/filter-realtime/overview. [Online; accessed 19-November-2017].

[15] Andy Piper. How much is 1 %, 2017. URL https://twittercommunity.com/t/how-much-is-1/95878. [Online; accessed 10-July-2018].

[16] Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. Prov-dm: The prov data model. 2012. URL http://www.w3.org/TR/prov-dm/.

[17] Interactive Advertising Bureau (IAB) Word of Mouth Marketing Association (WOMMA) American Association of Advertising Agencies (4A's). Mrc social media measurement guidelines. I:7–9, November 2015. ISSN 1541-1672.

[18] Fabián Riquelme and Pablo González-Cantergiani. Measuring user influence on twitter: A survey. *Information Processing and Management*, 52(5):949 – 975, 2016. ISSN 0306-4573. doi: 10.1016/j.ipm.2016.04.003.

[19] Lamjed Ben Jabeur, Lynda Tamine, and Mohand Boughanem. Active microbloggers: Identifying influencers, leaders and discussers in microblogging networks. In *String Processing and Information Retrieval*, pages 111 – 117, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34108-3. doi: 10.1007/978-3-642-34109-0_12.

[20] Gerasimos Razis and Ioannis Anagnostopoulos. Influencetracker: Rating the impact of a twitter account. volume abs/1404.5239, 2014. ISBN 978-3-662-44721-5. doi: 10.1007/978-3-662-44722-2_20.

[21] Qindong Sun, Nan Wang, Yadong Zhou, Hanqin Wang, and Liansheng Sui. Modeling for user interaction by influence transfer effect in online social networks. pages 486–489, Sept 2014. ISSN 0742-1303. doi: 10.1109/LCN.2014.6925823.

[22] Zhiguo Zhu, Jingqin Su, and Liping Kong. Measuring influence in online social network based on the user-content bipartite graph. *Computers in Human Behavior*, 52:184 – 189, 2015. ISSN 0747-5632. doi: 10.1016/j.chb.2015.04.072.

[23] Yi-Cheng Chen, Yi-Hsiang Chen, Chia-Hao Hsu, Hao-Jun You, Jianquan Liu, and Xin Huang. Mining opinion leaders in big social network. pages 1012–1018, March 2017. ISSN 1550-445X. doi: 10.1109/AINA.2017.147.

[24] Yupu Ding, Xiaoqing Yu, and Jing Lu. Application of k-means clustering algorithm in sina microblog. In *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*, pages 370–372, Aug 2013. ISBN 978-1-84919-707-6. doi: 10.1049/cp.2013.2038.

[25] Anton Borg, Fredrik Erlandsson, Piotr Brodka, and Henric Johnson. Finding influential users in social media using association rule learning. *CoRR*, abs/1604.08075, 2016. ISSN 1099-4300. doi: 10.3390/e18050164.

[26] Xiaoli Lin and Wei Han. Opinion leaders discovering in social networks based on complex network and dbscan cluster. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 292–295, Aug 2015. ISBN 978-1-4673-6593-2. doi: 10.1109/DCABES.2015.80.

[27] Said Akbar and Naeem Khan. Critical analysis of density-based spatial clustering of applications with noise (dbscan) techniques. 7:17–28, 10 2014. ISSN 2005-4270. doi: 10.14257/ijdta.2014.7.5.02.

[28] Nida Saddaf Khan, Maira Ata, and Quratulain Rajput. Identification of opinion leaders in social network. In *2015 International Conference on Information and Communication Technologies (ICICT)*, pages 1–6, Dec 2015. ISBN 978-1-4673-8907-5. doi: 10.1109/ICICT.2015.7469483.

[29] Jiang Wei, Gao Mengdi, Wang Xiaoxi, and Wu Xianda. A new evaluation algorithm for the influence of user in social network. *China Communications*, 13(2):200–206, Feb 2016. ISSN 1673-5447. doi: 10.1109/CC.2016.7405737.

[30] Wenlong Chen, Shaoyin Cheng, Xing He, and Fan Jiang. Influencerank: An efficient social influence measurement for millions of users in microblog. In *2012 Second International Conference on Cloud and Green Computing*, pages 563–570, Nov 2012. ISBN 978-1-4673-3027-5. doi: 10.1109/CGC.2012.31.

[31] Behnam Hajian and Tony White. Modelling influence in a social network: Metrics and evaluation. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 497–500, Oct 2011. ISBN 978-1-4577-1931-8. doi: 10.1109/PASSAT/SocialCom.2011.118.

[32] Long Ziyi, Cheng Fu Sui Jing, Sun Donghong, and Huang Yongfeng. Research on methods to identify the opinion leaders in internet community. pages 934–937, May 2013. ISSN 2327-0586. doi: 10.1109/ICSESS.2013.6615459.

[33] Khadije Rahimkhani, Abolfazl Aleahmad, Maseud Rahgozar, and Ali Moeini. A fast algorithm for finding most influential people based on the linear threshold model. *Expert Systems with Applications*, 42(3):1353 – 1361, 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2014.09.037.

[34] Huanhuan Liu, Xiaoqing Yu, and Jing Lu. Identifying top-n opinion leaders on local social network. In *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*, pages 325–328, Aug 2013. ISBN 978-1-84919-707-6. doi: 10.1049/cp.2013.1970.

[35] Alexandros Labrinidis and H. V. Jagadish. Challenges and opportunities with big data. *Proc. VLDB Endow.*, 5(12):2032–2033, August 2012. ISSN 2150-8097. doi: 10.14778/2367502.2367572.

[36] Aggelos Biboudis, Nick Palladinos, and Yannis Smaragdakis. Clash of the lambdas. *CoRR*, abs/1406.6631, 2014.

[37] Austin Appleby. Smhasher and murmurhash3 webpage, 2016. URL https://github.com/aappleby/smhasher. [Online; accessed 10-July-2018].

[38] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology*, 2(1):113–120, 1972. doi: 10.1080/0022250x.1972.9989806.

[39] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. ISSN 00380431.

[40] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, tools, and case studies. In *Synthesis Lectures on Data Mining and Knowledge Discovery*, volume 3, page 207, 10 2012. ISBN 9781608451166. doi: 10.2200/S00449ED1V01Y201209DMK006.

[41] Neo4j. Graph visualization for neo4j: Tools, methods and more, 2016. URL https://neo4j.com/developer/guide-data-visualization/. [Online; accessed 10-November-2016].

[42] Graph Lab. Visualization sgraph: methods graphlab.sgraph.show, 2016. URL https://turi.com/products/create/docs/generated/graphlab.SGraph.show.html#graphlab.SGraph.show. [Online; accessed 10-May-2016].

[43] Drew Skau. Why d3.js is so great for data visualization, 2013. URL https://visual.ly/blog/why-d3-js-is-so-great-for-data-visualization/. [Online; accessed 07-July-2018].

[44] Linkurio.us. Linkurious enterprise product page, 2017. URL https://linkurio.us. [Online; accessed 07-July-2018].

[45] Andrei Kashcha. Vivagraphjs, 2014. URL https://github.com/anvaka/VivaGraphJS. [Online; accessed 8-July-2017].

[46] Mongo DB. The mongodb 3.2 manual, 2016. URL https://docs.mongodb.com/manual/. [Online; accessed 8-July-2016].

[47] The Linux Foundation. Janusgraph distributed graph database, 2017. URL http://janusgraph.org/. [Online; accessed 07-July-2018].

[48] Lidson Jacob, Esteban Clua, and Daniel de Oliveira. Oh gosh!! why is this game so hard? identifying cycle patterns in 2d platform games using provenance data. *Entertainment Computing*, 19(Complete):65–81, 2017. ISSN 1875-9521. doi: 10.1016/j.entcom.2016.12.002.

[49] David Corsar, Peter Edwards, John Nelson, Chris Baillie, Konstantinos Papangelis, and Nagendra Velaga. Linking open data and the crowd for real-time passenger information. *Web Semantics: Science, Services and Agents on the World Wide Web*, 43:18 – 24, 2017. ISSN 1570-8268. doi: 10.1016/j.websem.2017.02.002.

[50] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, 2017. ISSN 1319-1578. doi: 10.1016/j.jksuci.2017.06.001.

[51] Reynold Xin, Joseph Gonzalez, Michael J. Franklin, and Ion Stoica. Graphx: a resilient distributed graph system on spark. In Peter A. Boncz and Thomas Neumann, editors, *GRADES*, page 2. CWI/ACM, 2013. ISBN 978-1-4503-2188-4. doi: 10.1145/2484425.2484427.

[52] Apache Spark Graphx. Graphx programming guide 2.2.0, 2017. URL https://spark.apache.org/docs/latest/graphx-programming-guide.html. [Online; accessed 15-August-2017].

[53] Clayton J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*. The AAAI Press, Jun 2014. ISBN 978-1-57735-659-2.

[54] Sancheng Peng, Aimin Yang, Lihong Cao, Shui Yu, and Dongqing Xie. Social influence modeling using information theory in mobile social networks. *Information Sciences*, 379:146–159, 2017. ISSN 0020-0255. doi: 10.1016/j.ins.2016.08.023.

[55] Yi-Cheng Chen, Wen-Yuan Zhu, Wen-Chih Peng, Wang-Chien Lee, and Suh-Yin Lee. Cim: Community-based influence maximization in social networks. *ACM Trans. Intell. Syst. Technol.*, 5(2):25:1–25:31, April 2014. ISSN 2157-6904. doi: 10.1145/2532549.

[56] Stuart P Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489.

[57] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. pages 161–172, 1999.

[58] Gengxin Sun and Sheng Bin. A new opinion leaders detecting algorithm in multi-relationship online social networks. *Multimedia Tools and Applications*, 77(4):4295–4307, Feb 2018. ISSN 1573-7721. doi: 10.1007/s11042-017-4766-y.

[59] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999. ISSN 0004-5411. doi: 10.1145/324133.324140.

[60] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. doi: 10.1162/jmlr.2003.3.4-5.993.

[61] Apache Spark. Apache spark mllib, 2016. URL https://spark.apache.org/mllib/. [Online; accessed 8-July-2016].

[62] Avner Barr. Murmurhash3 scala, 2018. URL https://gist.github.com/avnerbarr/82fb20d25c9b040f2caca8cdcefe7fc2/. [Online; accessed 8-July-2018].

[63] Martin Odersky, Stéphane Micheloud, Nikolay Mihaylov, Michel Schinz, Erik Stenman, Matthias Zenger, and et al. An overview of the scala programming language, 2004.

[64] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 599–613, Berkeley, CA, USA, 2014. USENIX Association. ISBN 978-1-931971-16-4.

[65] Andrei Kashcha. Rendering grid graph with 12 libraries, 2014. URL https://www.youtube.com/watch?v=Ax7KSQZ0_hk. [Online; accessed 8-July-2017].

Apéndices A

Apéndice I: Atributos de objetos de Twitter

Atributo	Tipo	Descripción
id	Long	Identificador único.
source	String	Medio utilizado para postear el tweet.
created_at	String	Fecha y hora UTC de creación del tweet.
text	String	Contenido del tweet en formato UTF-8.
lang	String	Identificador de idioma del contenido del tweet.
retweet_count	Int	Número de veces que el tweet ha sido retuiteado.
favorite_count	Int	Número de veces que el tweet ha sido del agrado de los usuarios.
in_reply_to_user_id	Long	Si el tweet se generó para responder a otro tweet, el atributo contiene el identificador del autor del tweet original.
in_reply_to_status_id	Long	Si el tweet se generó para responder a otro tweet, el atributo contiene el identificador del tweet respondido.
quoted_status_id	Long	Si el tweet se generó para citar otro tweet, el atributo contiene el identificador del tweet citado.
current_user_retweet	Int	Identificador del tweet del usuario retuiteado por el mismo usuario.
retweeted_status	Tweet	Si el tweet se generó por retuitear otro tweet, el atributo contiene el objeto del tweet retuiteado.
quoted_status	Tweet	Si el tweet se generó para citar otro tweet, el atributo contiene el objeto del tweet citado.

Tabla A.1: Atributos del objeto tweet de Twitter.

Atributo	Tipo	Descripción
id	Long	Identificador único de la cuenta de usuario.
name	String	Nombre de la cuenta de usuario, no necesariamente un nombre real.
created_at	String	Fecha y hora UTC de creación de la cuenta del usuario.
location	String	Localización de la cuenta de usuario, no necesariamente una localización real.
followers_count	Int	Número de usuarios que siguen al usuario.
friends_count	Int	Número de usuarios que sigue el usuario.

Tabla A.2: Atributos del objeto usuario de Twitter.

Apéndices B

Apéndice II: Estructura de tweet en formato JSON


```
{ "nodes": {
"1713483902": { "type": "User", "id": "1713483902", "color": "#FFED6F", "score_influence": "0.24439999999999998"},
"1478872944": { "type": "User", "id": "1478872944", "color": "#FFED6F", "score_influence": "0.14289999999999997"},
"-242839168": { "type": "Tweet", "id": "-242839168", "color": "#8DD3C7", "score_influence": "0.0"},
"1939829816": { "type": "Tweet", "id": "1939829816", "color": "#8DD3C7", "score_influence": "0.0"},
"1502964844": { "type": "Tweet", "id": "1502964844", "color": "#8DD3C7", "score_influence": "0.0"},
"-1993660816": { "type": "Tweet", "id": "-1993660816", "color": "#8DD3C7", "score_influence": "0.0"},
"964536886": { "type": "User", "id": "964536886", "color": "#FFED6F", "score_influence": "0.0"},
"343124356": { "type": "Tweet", "id": "343124356", "color": "#8DD3C7", "score_influence": "0.24439999999999998"},
"-334590988": { "type": "User", "id": "-334590988", "color": "#FFED6F", "score_influence": "0.0"},
"-1061745049": { "type": "User", "id": "-1061745049", "color": "#FFED6F", "score_influence": "0.0"},
"-2015112940": { "type": "Tweet", "id": "-2015112940", "color": "#8DD3C7", "score_influence": "0.0"},
"522554066": { "type": "Tweet", "id": "522554066", "color": "#8DD3C7", "score_influence": "0.14289999999999997"},
"741199852": { "type": "User", "id": "741199852", "color": "#FFED6F", "score_influence": "0.0"},
"-1953014872": { "type": "User", "id": "-1953014872", "color": "#FFED6F", "score_influence": "0.0"}
},
"links": [
{ "from": "1502964844", "to": "522554066", "type": "WasDerivedFrom"},
{ "from": "522554066", "to": "1478872944", "type": "WasAttributeTo"},
{ "from": "522554066", "to": "343124356", "type": "WasDerivedFrom"},
{ "from": "-242839168", "to": "964536886", "type": "WasAttributeTo"},
{ "from": "1502964844", "to": "741199852", "type": "WasAttributeTo"},
{ "from": "-1993660816", "to": "522554066", "type": "WasDerivedFrom"},
{ "from": "-242839168", "to": "522554066", "type": "WasDerivedFrom"},
{ "from": "-1993660816", "to": "-334590988", "type": "WasAttributeTo"},
{ "from": "1939829816", "to": "522554066", "type": "WasDerivedFrom"},
{ "from": "-2015112940", "to": "-1953014872", "type": "WasAttributeTo"},
{ "from": "343124356", "to": "1713483902", "type": "WasAttributeTo"},
{ "from": "-2015112940", "to": "522554066", "type": "WasDerivedFrom"},
{ "from": "1939829816", "to": "-1061745049", "type": "WasAttributeTo"}
]
 }
```


FIGURA B.1: Ejemplo de formato JSON para VivagraphJS.
Fuente: Elaboración propia.