UNIVERSIDAD NACIONAL AUTONÓMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Análisis de opinión en Twitter para *Amazon Go* a través de arquitectura Big Data"

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

ING. JUAN JOSÉ LÓPEZ MARTÍNEZ

DIRECTORA:

DRA. MARÍA DEL PILAR ÁNGELES

POSGRADO DE INGENIERÍA

Ciudad de México, 18 de agosto de 2019

Resumen

En este trabajo se aborda el análisis de la polaridad en las opiniones expresadas en Twitter, así mismo se identifican las inconformidades más frecuentes expresadas por los usuarios a través de este medio. Este análisis se basa en la identificación de tweets con contenido poco favorable sobre una novedosa propuesta traída a la realidad por el grupo comercial *Amazon*, la cual consistió en la creación de un nuevo tipo de supermercados en los cuales, a través de sensores, se tiene un control de los productos que el cliente lleva en su carrito de compras. Mientras el cliente abandona el establecimiento, a través de la cuenta del cliente, el sistema realiza el cobro de manera automática. Esta innovadora propuesta se llama *Amazon Go*, y su primera sucursal abrió sus puertas al público en 2006 en la ciudad de Seattle, Washington, USA.

A través de la redes sociales, en particular en Twitter, los usuarios externan sus opiniones respecto a los productos y servicios que consumen, y es a través de este medio que se ha logrado detectar insatisfacción por parte de los clientes respecto al servicio ofrecido por el grupo comercial. Esto es una problemática ya que es un riesgo latente para la pérdida de clientes.

Para llevar a cabo dicho análisis es crucial tomar en consideración diversos aspectos que se encuentran involucrados en las fuentes de información, tales como la velocidad a la que son generadas las publicaciones, la variedad que existe en la información y la cantidad de datos que son generados día a día.

Partiendo de esto se propondrá una solución que atienda la naturaleza de la problemática identificada, dicha propuesta buscará a través de los métodos de la Minería de Datos obtener un modelo descriptivo y uno predictivo.

Cada uno de los modelos que se presentan al final de este trabajo tiene como objetivo primordial aportar conocimiento de interés para el grupo comercial, además de brindar

soporte para la toma de decisiones y que permita al grupo comercial la oportunidad de que tomar acciones preventivas evitando poner en riesgo el capital que se ha invertido en el servicio *Amazon Go*.

Para implementar los modelos deseados se emplean tres herramientas tecnológicas: Hadoop, Python y RapidMiner, cada una con sus particularidades. La manipulación de cada una de estas herramientas es independiente, por lo cual se realizará una múltiple implementación del proceso analítico. Lo anterior tiene como objetivo identificar aquella herramienta tecnológica que ofrezca mayor eficiencia y mejor rendimiento en el desarrollo de las tareas requeridas para la obtención de los modelos deseados y así mismo seleccionar a aquellos modelos que presenten un mejor desempeño de acuerdo con los objetivos de esta tesis.

Para llegar a la identificación de las problemáticas externadas por los clientes se obtiene un corpus conformado por tweets cuyo contenido está directamente relacionado con *Amazon Go*. Debido a las características de la fuente de información (Twitter) no es viable analizar directamente los datos, por lo cual es necesario recurrir a un proceso de limpieza y adecuación de los datos.

Teniendo listo el corpus se implementa un análisis sentimental con dos objetivos principales: el primero de ellos consiste en conocer la distribución de la polaridad de las opiniones de los usuarios que usan la red social; y, el segundo es poder segmentar los tweets, dejando únicamente aquellos que han sido identificados con un sentimiento negativo, ya que de esta forma se facilita la identificación de los problemáticas que expresan los usuarios de Twitter.

Mediante la ejecución del algoritmo de agrupamiento con lógica difusa se forman grupos conformados por los tweets que han sido clasificados negativamente, y a partir de esta segmentación se identifican las características que tienen los miembros de cada grupo, logrando así identificar las principales inconformidades externadas por los clientes, además de que también se obtiene información de interés para el grupo el comercial *Amazon*.

Agradecimientos

A MI FAMILIA

Mis padres, de quienes siempre he contado con su apoyo incondicional. A mi hermana, quien me ha motivado a inspirado a estudiar un posgrado.

A LA UNAM

Por darme oportunidad de ser parte de ella, por los conocimientos y valores que he adquirido y me han ayudado en mi formación personal y profesional.

A LA DRA. MARÍA DEL PILAR ÁNGELES

Directora de esta tesis, por el apoyo, la asesoría y la paciencia brindada durante el desarrollo de la misma.

A MIS SINODALES

Quienes me brindaron su opinión de este trabajo, y gracias a ello pude mejorar aspectos que no había percibido.

A CONACYT

Por la beca otorgada durante mis estudios en la maestría, la cual fue parte primordial para que pudiera concluir mis estudios de manera satisfactoria.

A MIS AMIGOS...

Olga, por compartir tantas tazas de café, así como el tiempo en el tráfico que pasamos juntos; Raúl, quien compartió sus experiencias y conocimientos en proyectos que realizamos juntos a lo largo de este posgrado; Eduardo, quien con su optimismo y masajes relajantes nos unía como compañeros creando un ambiente más agradable; Miguel, quien ofreció siempre su ayuda de manera incondicional y en quien encontré gran apoyo para el manejo de nuevas tecnologías; Carlos, Ricardo, Julieta, Mariana, Gibran, Daniel, Rosalía y Antonio, en quienes encontré su invaluable amistad, así como su apoyo y motivación a lo largo de nuestra estancia en el posgrado.

Lista de acrónimos

API Application Programming Interface.
ASCII American Standard Code for Information Interchange.
BD Base de Datos.
BI Business Intelligence.
BSP Bulk Synchronous Parallel.
CSV Comma-Separated Values.
DBMS Data Base Management System.
DW Data Warehouse.
ETL Extract, Transform and Load.
HDFS Hadoop Distributed File System.
IoT Internet of Things.
L/E Lectura y Escritura.
NLTK Natural Language Toolkit.
OLAP On-Line Analytical Processing.
OLTP OnLine Transaction Processing.
RDBMS Relational Data Base Management System.
ROI Return On Investment.
SAP System Analysis and Program Development.
SQL Structured Query Language.
YARN Yet Another Resource Negotiator.

Índice general

Índice de figuras

Índice de tablas

I think the brain is essentially a computer and consciousness is like a computer program. It will cease to run when the computer is turned off. Theoretically, it could be re-created on a neural network, but that would be very difficult, as it would require all one's memories.

Stephen Hawking

CAPÍTULO

1

Introducción

1.1 Contexto

En las últimas décadas se ha presentado un cambio en el almacenamiento y procesamiento de la información de las empresas, se ha pasado de un almacenamiento en documentos de papel a uno digital. Actualmente las empresas almacenan grandes volúmenes de datos en clústeres de discos o en la nube. Esta creciente demanda para el almacenamiento y procesamiento masivo de datos es consecuencia de las actividades cotidianas del negocio. De tal forma que las tareas de generación, almacenamiento y manipulación de todo tipo de datos se tornan cada vez más complejas.

En años recientes el análisis de la información almacenada en grandes bases de datos se ha convertido en una importante herramienta para las empresas. A través de la preparación, el sondeo y la exploración de los datos, es posible obtener como resultado final información *oculta*, y de gran utilidad (comportamientos y futuras tendencias) que brinde soporte para la toma de decisiones y permita hacer proyecciones a futuro para el negocio.

Para llevar a cabo este análisis se recurre a fuentes de datos internas, externas y a los métodos y algoritmos de la *Minería de Datos*, así como otras herramientas con el propósito de obtener un modelo que cumpla con los objetivos de negocio.

En el contexto actual, las redes sociales se han convertido en un medio a través del cual los usuarios externan sus opiniones respecto a los servicios y/o productos que consumen. Por medio de esta plataforma las empresas pueden obtener información de gran relevancia para el negocio, en especial por el hecho de tratarse de información que proviene de primera fuente, es decir, del cliente mismo.

En esta tesis, se pretende aprovechar esta nueva fuente de información para el grupo comercial *Amazon*, el cual es una compañía estadounidense de comercio electrónico y servicios de cómputo en la nube. Actualmente, cuenta con sedes en diversos países logrando de esta manera comercializar productos a nivel internacional.

Amazon busca mantenerse a la vanguardia ofreciendo nuevos servicios que se caractericen por ser innovadores y satisfagan de manera óptima las necesidades de sus clientes. Derivado de este paradigma, *Amazon* a finales del 2016 presentó una propuesta para la construcción de tiendas físicas con puntos de entrega en las aceras ofreciendo principalmente productos alimenticios, esta idea es llamada *Amazon Go*. Meses después, se inauguró la primera sucursal de *Amazon Go* en la ciudad de Seattle, Washington, USA.

La parte innovadora de esta propuesta consiste en la promesa de la erradicación de las filas para pagar que encontramos en cualquier supermercado; a través del uso de una variedad de sensores se actualiza el carrito de compras, realizando el cobro automáticamente a la cuenta *Amazon* del cliente en el momento en el que éste deja la tienda. por lo tanto, no hay filas para realizar el pago.

Los usuarios a través de la red social Twitter[1], difunden su opinión sobre el servicio ofrecido por *Amazon Go*, mediante el uso de diversos hashtags entre los cuales destaca el hashtag *#amazongo*. Considerando que la información proviene de la red social se entiende que ésta carece de una estructura explícita. Además, como estos mensajes son generados en lapsos de tiempo muy pequeños, se requerirá tener una alta capacidad para poder capturar, almacenar y procesar el flujo de información. Debido a que la información se genera en periodos de tiempo tan cortos se obtendrán grandes volúmenes de datos que continuamente incrementarán su tamaño.

[1] http://twitter.com

1.2 Problemática

Lamentablemente, la innovadora propuesta de *Amazon* se trajo a la realidad parcialmente, la imagen de un supermercado sin filas solo duró hasta su apertura al público, y como consecuencia de esto, a través de las redes sociales los clientes comenzaron a compartir sus inconformidades con el servicio. *Amazon* debe enfocarse en identificar las razones por las cuales su propuesta no está siendo aceptada completamente por los usuarios, ya que estas opiniones negativas ponen en riesgo la inversión que el grupo empresarial ha realizado. Pos esta razón es necesario recurrir a un proceso analítico que permita tener conocimiento de la situación actual y así poder tomar acciones correctivas.

1.2.1 Pérdidas monetarias

La insatisfacción del cliente se traduce directamente en términos monetarios como cifras negativas para la empresa. Es un riesgo latente para la inversión que el grupo comercial *Amazon* ha realizado al poner en operación este tipo de supermercados.

Cabe mencionar que actualmente no se cuenta con un informe de balance financiero público en el cual se vea reflejada la situación actual del negocio.

1.2.2 Pérdidas de clientes

Las inconformidades expresadas a través de las redes sociales advierten sobre el riesgo de pérdida de clientes, ya que al no cumplir con las expectativas de los clientes se encuentra latente la posibilidad de que ellos opten por otro servicio que se adecúe más a sus demandas o necesidades.

Al final del día este fenómeno también se puede expresar como pérdidas económicas para la empresa, repercutiendo directamente en la inversión que ha realizado el grupo comercial.

1.3 Hipótesis

A través de la información que comparten los usuarios en Twitter es posible identificar las causas de insatisfacción de los clientes a través de un análisis predictivo y descriptivo sobre un ambiente analítico Big Data.

1.4 Objetivos

Objetivo general: Identificar y analizar las opiniones compartidas por los usuarios de *Amazon Go* en Twitter, de tal forma que dicho análisis permita al grupo comercial tener soporte para toma decisiones, y así buscar generar mayor satisfacción en sus clientes.

Objetivos específicos:

1. Obtener información no estructurada de la red social Twitter, relacionada con opiniones negativas sobre *Amazon Go*.
2. Obtener un modelo analítico descriptivo a través del cual se identifique la opinión de los clientes expresada a través de Twitter.
3. Obtener un modelo predictivo que permita analizar la información que se genera en Twitter.
4. Implementar el proceso analítico en diversas herramientas tecnológicas con la finalidad de comparar el desempeño de estas.

1.5 Aproximación

Con los datos obtenidos a partir de Twitter es viable llevar a cabo el análisis, el cual será predictivo y descriptivo (tal como se explicará en el capítulo 3 de este trabajo), así mismo este será de utilidad para la empresa dando soporte a la toma de decisiones.

Posteriormente, se considera relevante desarrollar una aplicación con una arquitectura de tipo Big Data, la cual permitirá llevar a cabo el análisis deseado con el propósito de obtener los modelos predictivo y descriptivo que brinden soporte para la toma de decisiones para el grupo empresarial.

Para la obtención de los datos a partir de las red social se tiene contemplado emplear python

como herramienta para la extracción de los tweets. Después de obtener todos los datos, éstos serán transferidos a un HDFS[1] para poder comenzar con la siguiente etapa que consistirá en la limpieza de los datos.

Posterior a la limpieza y preparación de los datos se estudiaran diversas metodologías para realizar un análisis de esta naturaleza. A partir de esta revisión se podrá determinar aquellas técnicas de *Minería de Datos* más adecuadas para abordar la identificación de opiniones negativas. Esto con el propósito de poder brindar información que sea de interés para el negocio, por ejemplo, saber qué tan satisfechos están los clientes con los bienes y/o servicios que han adquirido, encontrar las principales dificultades o decepciones a las que se enfrentan sus clientes, conocer las nuevas necesidades de los clientes, etc.

Finalmente, se evaluarán los resultados obtenidos y se verificará qué tan útil y eficiente es el conocimiento resultante del proceso analítico, de ser necesario se volverán a aplicar las técnicas analíticas con las modificaciones necesarias y de ser requerido, cambiar las estrategias a través de las cuales se está llevando a cabo el análisis de la información.

1.6 Contribuciones

A través de la aplicación de los métodos de la *Minería de Datos* se busca obtener un modelo de tipo predictivo y uno descriptivo. Para poder llegar a estos modelos, se deberá emplear una serie de pasos que serán establecidos por una metodología afín a la naturaleza del problema a abordar.

Además, la solución que se propone inicialmente se enfoca a una red social en particular, pero esto realmente no es una limitante, ya que para trabajo futuro puede escalarse a más redes sociales, y la aplicación analítica podrá continuar operando de manera satisfactoria.

Por otro lado, con la realización de este trabajo se propondrá una arquitectura analítica Big Data. A través de la cual se logrará identificar opiniones negativas de algún tema en particular, así como la identificación de las características que comparten en común estas opiniones; lo anterior permitirá identificar patrones y tendencias existentes en los comentarios.

[1] Es un sistema de archivos distribuido, escalable y portátil escrito en Java y creado especialmente para trabajar con archivos de gran tamaño.

A partir de esta identificación se podrán tomar acciones correctivas, y así mismo, dar seguimiento a dichas opiniones con el objetivo de reducir el número de comentarios negativos, y en consecuencia generar mayor satisfacción en el cliente.

Algo que caracterizará esta arquitectura es que no excluirá a nuevas opiniones negativas que vayan surgiendo a partir de nuevas publicaciones, ya que al ser un proceso iterativo, al analizar las nuevas opiniones es posible llevar a cabo la identificación de nuevos tópicos.

Finalmente, en las contribuciones concernientes al grupo empresarial, se buscará dar respuesta a preguntas que podrán ser de utilidad para la toma de decisiones.

Las preguntas que se desean resolver al final de este trabajo son:

Preguntas de tipo descriptivas:

- ✓ ¿Con base en el comportamiento de las opiniones de los clientes, existe viabilidad de abrir nuevas sucursales?
- ✓ ¿Los usuarios están satisfechos con la propuesta *Amazon Go*?
- ✓ ¿Qué tipo de quejas son las que se presentan con mayor frecuencia?
- ✓ ¿Cuáles son las sugerencias expresadas por los clientes?
- ✓ ¿Cuál es la distribución de comentarios positivos y negativos?

Preguntas de tipo predictivas:

- ✓ ¿Cómo se comportará la popularidad de *Amazon Go* en Twitter?
- ✓ ¿Cómo será el comportamiento de las opiniones de los clientes de *Amazon Go* en Twitter a futuro?
- ✓ ¿Cuál será la popularidad de los tweets con contenido desfavorable para la empresa?
- ✓ ¿Cuáles son los productos más populares dentro de las opiniones de *Amazon Go* en Twitter?
- ✓ ¿Es posible predecir un crecimiento en la difusión de los tweets con contenido positivo sobre *Amazon Go*?

1.7 Limitaciones

Para la realización de este trabajo se identifican las siguientes limitantes:

1. La red social que se empleará para el análisis es Twitter.
2. El tamaño del corpus estará limitado a la capacidad de procesamiento y almacenamiento del hardware y software a través del cual se obtengan los datos y donde se realiza el análisis de los mismos.
3. Se obtendrá y analizará información publicada dentro del periodo comprendido de febrero a agosto del 2018.
4. Se requerirá utilizar el framework de Apache Hadoop, así como RapidMiner y el lenguaje Python para realizar las tareas de almacenamiento, limpieza y análisis de los datos.

1.8 Estructura de la tesis

El resto de esta tesis se encuentra estructurado de la siguiente forma:

- **Capítulo 2: Problemática actual de los sistemas de información**
  Se aborda un breve panorama de los sistemas de información y las nuevas tendencias en el almacenamiento de los datos de acuerdo con las nuevas demandas.

- **Capítulo 3: Marco teórico**
  Se presenta de manera formal *Big Data*, haciendo mención de cuales son los fenómenos que dan origen a este término y cómo es abordado en la actualidad.

- **Capítulo 4: Metodologías de desarrollo de aplicaciones de tipo Big Data**
  Después de que se ha introducido la temática de *Big Data* de una manera formal, en este capítulo se presentarán las metodologías de desarrollo de aplicaciones de ambiente *Big Data*, esto se lleva a cabo a través de un breve estudio de diversos trabajos realizados por expertos en el área.

- **Capítulo 5: Arquitectura solución y Metodología utilizada para análisis Big Data**
  Con base en lo que se presenta en el capítulo 4, en este capítulo se introduce la elección de la metodología de desarrollo para el modelo que se desea obtener en el presente trabajo. Aquí mismo se introduce la arquitectura que será uilizada para implementar la aplicación analítica en cuestión.

- **Capítulo 6: Implementación de la propuesta**
  En este capítulo se desarrollan cada una de las fases que comprenden la metodología de desarrollo elegida, de tal forma que al concluir este capítulo se obtendrán los modelos analíticos de interés así como sus respectivas evaluaciones con las que se establecerán criterios de aceptación.

- **Capítulo 7: Análisis de resultados y conclusiones**
  En el capítulo 7 se discuten y analizan los resultados que se obtuvieron de la implementación de los procesos analíticos. Se externan recomendaciones y posibles modificaciones a futuro para continuar con el trabajo desarrollado en esta tesis.

Machine consciousness refers to attempts by those who design and analyse informational machines to apply their methods to various ways of understanding consciousness and to examine the possible role of consciousness in informational machines.

Igor Aleksander

CAPÍTULO

2

Problemática actual de los sistemas de información

2.1 Sistemas de archivos

A lo largo de la historia de las tecnologías de la información se han presentado diversas alternativas como respuesta a las problemáticas ocurrentes en cada contexto, por ejemplo, para la necesidad de almacenar la información que era procesada a través de los sistemas de cómputo, surgieron los sistemas de archivos, los cuales cumplían con su objetivo (almacenar y recuperar información).

Con los sistemas de archivos el operador del sistema era quien manipulaba directamente el archivo, funcionaban bien cuando el número de elementos almacenados era pequeño [1], pero presentaban diversas desventajas, principalmente se distinguían por ser realmente ineficientes en diversos aspectos tales como concurrencia, integridad y durabilidad de la información.

2.2 Sistemas Manejadores de Bases de Datos

Dadas la dificultades que presentaban los sistemas de archivos, y buscando satisfacer aquellos aspectos que no podían ser cumplidos por los sistemas de archivos surgen los sistemas manejadores de bases datos (DBMS por sus siglas en inglés), junto con los cuales surgen diversos modelos[2] como son el modelo jerárquico, el modelo de red, hasta llegar al modelo relacional el cual hoy en día sigue siendo una excelente alternativa para las operaciones transaccionales ya que este modelo se encuentra basado en la teoría de conjuntos, lo cual le brinda un gran soporte teórico al modelo.

A través de estos sistemas manejadores de bases de datos el proceso de almacenamiento de datos se convierte una tarea sencilla, además de que los aspectos que no podían ser atendidos por los sistemas de archivos aquí son una tarea primordial. Con el uso de los DBMS en los sistemas de información se comienzan a identificar dos tipos de procesamiento: OLTP y OLAP.

2.2.1 Sistemas de tipo OLTP

Los sistemas de tipo OLTP se caracterizan por contar con un modelo adecuado para atender las peticiones transaccionales, de acuerdo con Inmon[3], agilizan el acceso a los datos, además de que brindan un panorama más amplio para el contexto de negocios.

Esto permitió que la computadora podiera ser utilizada para tareas que antes no eran posibles, tales como sistemas que permitan manejar reservaciones, sistemas de control de manufacturación, etc.

Se identifican como transacciones todas aquellas operaciones que realizan algún movimiento en la base de datos: creación, consulta, eliminación y actualización de datos.

Retomando el ejemplo de un sistema que maneja reservaciones de vuelos, se identifican diversas transacciones, por ejemplo, cuando un usuario se registra para realizar una compra, el sistema ejecuta una escritura en la base de datos; cuando el usuario desea consultar los lugares disponibles de un vuelo, el sistema manda una petición de lectura a la base de datos; cuando el usuario selecciona su asiento y confirma la compra se realiza una escritura en la base de datos; cuando se cancela algún vuelo se realiza una eliminación en la base de datos.

Este ejemplo, considerando que es un sistema que está en producción, realizará las transacciones mencionadas de manera continua debido que se tendrán diversos clientes de manera simultánea,

siendo esta la razón por la cual se deberá tener un modelo de datos OLTP que pueda hacer frente a estas demandas.

Con este tipo de sistemas el procesamiento de transacciones en línea tiene claramente dos beneficios: *simplicidad* y *eficiencia*. Por un lado la *simplicidad* se ve reflejada a través de las consultas que se tienen que ejecutar para realizar una transacción, brindando facilidad de operación de este tipo de sistemas por parte de los clientes. Mientras que la *eficiencia* se aprecia gracias a que los procesos individuales se ejecutan mucho más rápido.

Sin embargo, los sistemas de OLTP conllevan cuestiones que deben de ser consideradas ya que pueden suponer un problema: *seguridad* y *costes* ya sean económicos o de tiempo. Debido a que un sistema OLTP brinda disponibilidad a todo el mundo a través de los servicios ofrecidos por las empresas, hacen que sus bases de datos sean más susceptibles a intrusos[4]. En lo que respecta al costo, este tipo de sistemas tienen el inconveniente de que una pequeña perturbación en el sistema puede causar una gran cantidad de problemas, que a su vez se puede entender como una pérdida de tiempo y dinero.

2.2.2 **Sistemas de tipo OLAP**

Los sistemas de tipo OLAP comenzaron a emplearse debido a la ineficiencia de los sistemas de tipo OLTP para cuestiones analíticas, especialmente cuando el volumen de datos aumentaba drásticamente. La razón principal fué que en ocasiones, las consultas analíticas[1] no podían ser atendidas debido a que los conjuntos de datos con los cuales se debía operar eran tan grandes que no cabían en la memoria.

De este modo se originan los almacenes de datos (DW) que fueron introducidos por Bill Inmon, quien los define como *"un repositorio de datos orientado a temas (los datos en la base de datos están organizados de manera que todos los elementos de datos relativos al mismo evento u objeto del mundo real queden unidos entre sí), variante en el tiempo (los cambios producidos en los datos a lo largo del tiempo quedan registrados para que los informes que se puedan generar reflejen esas variaciones), no volátil (la información no se modifica ni se elimina, una vez almacenado un dato, éste se convierte en información de sólo lectura, y se mantiene para futuras consultas) e integrado*

[1]Es un tipo consulta que suele involucrar a datos obtenidos a partir de otras consultas, además pueden contener funciones de agregación y agrupación, tales como promedio, suma, el máximo, el mínimo, entre otros.

(la base de datos contiene los datos de todos los sistemas operacionales de la organización, y dichos datos deben ser consistentes)"[3].

En la figura 2.1 se presenta el diseño de forma genérica de un Data Warehouse, en la parte izquierda se encuentran las fuentes de datos que pueden ser internas o externas, en este ejemplo se tienen datos almacenados por su zona geográfica. A través de diversas herramientas se extraen los datos de todas las fuentes, se adecúan y se cargan en el Data Warehouse (siendo en este punto donde los datos se vuelven no volátiles) para que puedan ser explotados a través de herramientas analíticas, obteniendo como resultado reportes que fungen como soporte para la toma de decisiones.

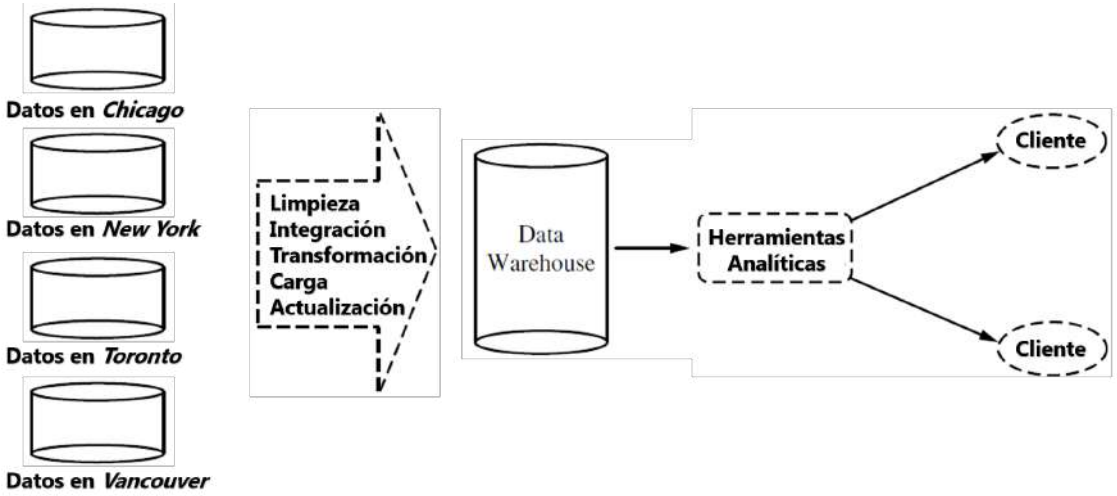


Figura 2.1: Framework de un Data Warehouse (extríada y traducida de [5]).

Los sistemas de tipo OLAP cuentan con un modelo de datos en el cual existe mucha redundancia de datos, pero gracias a esto se logra reducir considerablemente los tiempos de respuesta para las consultas de tipo analíticas. Como consecuencia del incremento de la redundancia de los datos, este tipo de sistemas no pueden ser utilizados para realizar consultas de tipo transaccionales, ya que el tiempo de ejecución de una consulta podría tomar demasiado tiempo debido a que si se modifica un dato, este cambio deberá de verse reflejado en todas las instancias en las que se encuentra almacenado.

En conclusión, OLTP y OLAP son tecnologías independientes y al mismo tiempo complementarias. OLTP permite realizar transacciones correspondientes a las actividades diarias derivadas del negocio, mientras que para que estas transacciones puedan ser analizadas tienen que ser llevadas a un sistema de tipo OLAP, para lo cual se requiere recurrir a las herramientas de tipo ETL, las

cuales unificarán las fuentes de datos y las almacenarán en el sistema de tipo OLAP, a través del cual es viable ejecutar consultas de tipo analíticas.

2.3 Bases de datos columnares

Buscando alternativas para satisfacer la demanda de capacidad de almacenamiento y procesamiento de grandes volúmenes de información en un solo sistema, surgen las bases de datos columnares, las cuales persiguen ofrecer tiempos de respuesta menores al fragmentar una entidad (en el contexto del modelo relacional) en columnas.

El almacenamiento basado en columnas para las tablas de bases de datos es un factor importante en el desempeño de las consultas analíticas, ya que reduce notablemente los requisitos globales de L/E del disco, y disminuyen el volumen de datos[1] que hay que cargar en memoria.

Para reducir el volumen de datos, en cada una de las entidades en la base de datos se recurre a la creación de un diccionario, el tamaño de cada diccionario será igual a la cardinalidad del atributo de la entidad para el cual se está creando dicho diccionario. En la tabla 2.1 se presenta una entidad correspondiente a las *fechas* almacenadas en una base de datos convencional. Esta tabla tiene un total de ocho instancias y una cardinalidad de tres.

Fecha
12/10/2007
12/10/2007
15/02/2011
15/02/2011
15/02/2011
15/02/2011
01/01/2017
01/01/2017

Tabla 2.1: Ejemplo de datos de tipo fecha, forma en que se encuentran en un RDBMS.

Para la creación del diccionario a partir de la tabla presentada en la tabla 2.1, se toman los valores únicos dentro del dominio del atributo, siendo que en este caso tendremos un diccionario

[1] A través de la comprensión de datos, lo cual de acuerdo con Lelewer y Hirschberg[6], reduce la redundancia en los datos almacenados.

conformado por tres valores, y a cada uno de los valores se les asigna una *llave* a través de la cual se identificará de manera única al valor al que hace referencia. En la tabla 2.2 se muestra el diccionario que se generó para los datos presentados en la tabla 2.1.

Fecha	Llave
12/10/2007	1
15/02/2011	2
01/01/2017	3

Tabla 2.2: Ejemplo de diccionario para compresión de datos en BD Columnar

Finalmente se hace un mapeo de los datos de la tabla original, siendo estos reemplazados por las llaves que se han generado en la creación del diccionario, de tal modo que la tabla final que se almacenará dentro del motor de base de datos columnar será la que se aprecia en la tabla 2.3

Fecha
1
1
2
2
2
2
3
3

Tabla 2.3: Ejemplo con datos comprimidos en BD Columnar

El motor de base de datos columnar obtiene ventaja de que en lugar de tener que almacenar un dato de tipo *date* (como en este ejemplo) o *varchar* las *n* instancias, únicamente será almacenada la llave en las *n* ocurrencias de dicho valor, lo cual ahorra espacio y se reduce el tiempo de respuesta. Ya que al tener índices de tipo numérico, cuando se realiza un consulta para buscar un valor el tiempo de respuesta es menor, pues la comparación de datos de tipo numérico es más eficiente que la comparación de otros tipos de datos, en consecuencia el tiempo de búsqueda para recuperar el atributo es menor[7].

Mientras que una base de datos relacional está optimizada para almacenar filas de datos (normalmente para aplicaciones transaccionales) una base de datos columnar está optimizada para lograr una recuperación rápida de columnas de datos, dejando a las bases de datos columnares como una opción para aplicaciones analíticas.

2.4 Nuevas demandas

Pareciera que con las soluciones existentes es posible solventar la demanda de la transaccionalidad y el análisis de los datos que son generados a través las operaciones dentro de cualquier aplicación. Sin embargo, las necesidades continúan cambiando, presentándose como principal problema el análisis de grandes bloques de datos (originados principalmente por *Internet de las Cosas (IoT)*) con gran variedad de datos (datos estructurados, semi o no estructurados) y con una demanda de tiempos de respuesta muy pequeños. En la figura 2.2 se ilustran factores que generan grandes cantidades de datos diariamente.

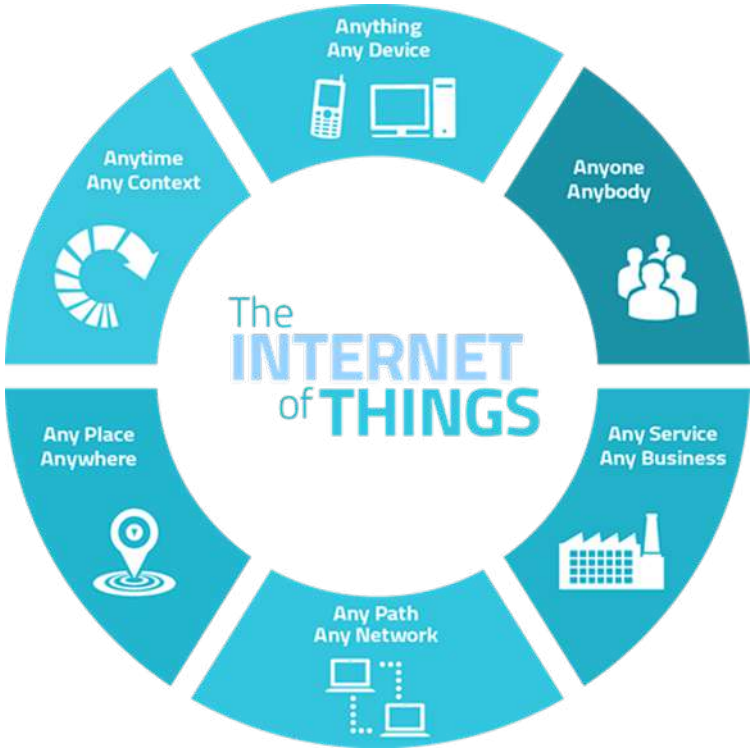


Figura 2.2: Componentes que conforman Internet de las Cosas. Extraída de [8]

Estas tres características mencionadas: volumen, velocidad y variedad de la información, conllevan a que las técnicas analíticas convencionales sean incapaces de hacer frente a las nuevas demandas, lo que permite el surgimiento de Big Data.

Las fuentes de información principales ya no son exclusivamente las bases de datos relacionales o archivos, ahora se suelen incluir redes sociales, archivos multimedia, páginas web, entre otras, que en conjunto implican que llevar a cabo un análisis de todas estas fuentes de datos unificadas se

convierta en una tarea realmente complicada, especialmente, como se ha mencionado, la complejidad radica en la manipulación de grandes volúmenes de datos, los cuales ahora no necesariamente están estructurados sino que además son generados en periodos de tiempo muy pequeños.

A partir de este nuevo contexto de Big Data es viable tomar ventaja de esta tecnología utilizándola en diversos sectores, tales como salud, educación, finanzas e industria. En el caso particular de la industria privada, Big Data ofrece la oportunidad de conocer la situación actual entre el negocio y su relación con los clientes, tener una concepción de los clientes (perfil de los clientes), comprender el contexto de las situaciones que orillan a que un cliente deje de utilizar un servicio, o bien, aquellas condiciones que permiten que existan nuevos consumidores. Conocer cuáles son los aspectos que les agradan o desagradan sobre los servicios y/o productos ofrecidos. Todo lo anterior permite contar con un gran soporte para la toma de decisiones, tales como saber si es viable expandir el negocio a nuevas áreas, hacer cambios en los productos, diseñar una mejor campaña de negocio, etcétera.

Computers are getting smarter and smarter. Scientists say that soon they will be able to talk to us (and by 'they' I mean computers, I doubt very much that scientists will be able to talk to us).

Dave Barry

CAPÍTULO

3

Marco teórico

3.1 Importancia del análisis de la información

El análisis de los datos, sin importar la manera en la que son generados, se ha convertido en una tarea de gran relevancia para cualquier empresa u organización, ya que la información que se puede obtener a través del análisis de dichos datos ha pasado a convertirse en uno de los activos de mayor importancia para las empresas y organizaciones, tal como menciona Alexandros Labrinidis *"existe un amplio reconocimiento del valor de los datos y de los beneficios obtenidos mediante su análisis"*[9], esto se debe básicamente al hecho de que a partir del análisis de los datos es posible describir y predecir comportamientos o hechos.

Uno de los principales usos del Big Data es crear valor a partir del análisis de grandes cantidades de información y para lograrlo depende de manera significativa en la capacidad analítica. La cuestión es preguntar correctamente a los datos de tal manera que la información recopilada pueda dar las respuestas necesarias para entender lo que sucede, por qué sucede y hasta lo que puede suceder.

3.2 Minería de Datos

Existen tres vertientes analíticas las cuales se basan en qué es lo que se desea conocer, estas áreas analíticas se conocen como análisis predictivo, descriptivo y prescriptivo[10].

El análisis descriptivo[11] se conoce comúnmente como herramientas de inteligencia de negocios (BI). La analítica descriptiva tiene en cuenta lo que ocurrió con la mejora de la toma de decisiones basada en las lecciones aprendidas. Este tipo de análisis permite tener un panorama sobre características que hay entre los datos. Además, es importante destacar que estas características de los datos suelen no ser perceptibles a simple vista, siendo esta la razón por la cual es necesario recurrir a técnicas especializadas para un análisis más profundo de los datos.

Otra vertiente, el análisis predictivo[12], consiste en realizar un análisis que permita obtener un modelo predictivo, este tipo de análisis es mayormente utilizado por las aseguradoras para evaluar lo que podría suceder al analizar el pasado para predecir los resultados futuros.

"Se busca optimizar el rendimiento de un sistema utilizando conjuntos de tecnologías inteligentes para descubrir las relaciones y patrones dentro de grandes volúmenes de datos para predecir eventos futuros"[13], esto quiere decir que a partir de los datos se tratan de predecir tendencias y probabilidades futuras, identificar patrones con el propósito de que conociendo los valores que pueden tomar unos atributos se logre predecir el valor de otro(s).

Finalmente el análisis prescriptivo[11] analiza los datos para encontrar cuál es la solución entre una gama de variantes. Su tarea es optimizar recursos y aumentar la eficiencia operativa. Usa técnicas de simulación y optimización, logrando señalar cuál es el camino que conviene realmente elegir.

En realidad, informa acerca de lo que debiera suceder buscando mejorar el resultado y proporcionando recomendaciones para maximizar indicadores de negocio. En realidad, *"tanto la predictiva como la prescriptiva atraerán, según diversos estudios, el 40 % de la nueva inversión en la empresa en Inteligencia de Negocios y Analítica de datos"*[14].

Aplicando estos tipos de análisis en el ámbito empresarial se pueden obtener grandes beneficios, en especial para las ventas de un producto o servicio, a partir de un proceso analítico de este tipo existe una gran probabilidad de poder identificar un próximo cliente, saber si una campaña publicitaria tendrá éxito, identificar cuando un cliente esté por dejar de serlo y tomar acciones tempranas ante esta situación.

3.2.1 **Análisis sentimental**

Un análisis sentimental de acuerdo con Pak y Paroubek[15] consiste en un análisis lingüístico, con el propósito de extraer información subjetiva del corpus[1].

Un análisis sentimental puede ser entendido como una tarea de clasificación masiva de un corpus, dicha clasificación se lleva a cabo mediante la polaridad del texto, y en un análisis básico existen dos vertientes, la primera es una clasificación binaria en donde se manejan dos únicas etiquetas (positivo y negativo), o bien, puede existir una tercera etiqueta la cual corresponderá a la neutralidad del mensaje (neutro). Sin embargo, en un análisis sentimental avanzado no se limita a la determinación de la polaridad del mensaje, en un análisis de este tipo se puede buscar otras características, por ejemplo, estados emocionales tales como alegría o tristeza.

Para la obtención un modelo que permita realizar la clasificación de la polaridad del texto existen diversas formas de generar dicho modelo, por un lado se encuentra una primera aproximación mediante una suma ponderada de los componentes lingüísticos del texto a partir de un diccionario, es decir, en el diccionario se tiene un conjunto de palabras que han sido seleccionados previamente y se les ha asignado una valor numérico dependiendo de la polaridad que expresa dicha palabra, por ejemplo, 'deficiente'tendrá una connotación negativa, por lo cual se le asocie un valor negativo. Para calcular la polaridad absoluta del mensaje se suman los valores de todas las palabras contenidas en el texto y a partir de este cálculo se categoriza al texto.

La ponderación de las palabras contenidas en el diccionario empleado es un heurístico del cual dependerá la determinación de la categoría, de tal forma que este enfoque se torna un poco subjetivo. Existen otras formas de obtener un analizador sentimental que utilizan la probabilidad, por ejemplo, Das y Chen[16] implementan diversos algoritmos clasificadores, tales como son el clasificador de Naive Bayes, clasificador vector distancia, clasificador basado en discriminación (discriminant-based), clasificador por adjetivos y adverbios (adjective-adverb phrase) y Bayes.

[1] Término empleado en lingüística para referirse un conjunto cerrado de textos o de datos destinado a la investigación científica.

3.2.1.1 **Evaluación de análisis sentimental: matriz de confusión**

La matriz de confusión es una tabla que muestra el rendimiento de un algoritmo de clasificación mediante la comparación del valor predicho de la variable de destino con su valor real. Para generar la matriz de confusión se realiza el conteo de las instancias que fueron clasificadas correcta e incorrectamente, esto para cada una de las posibles etiquetas.

Cada columna de la matriz representa uno de los posibles valores que puede tomar la variable destino, mientras que cada renglón representa las observaciones en una clase real (o viceversa). Por consecuente, esta técnica puede ser empleada para cuantificar el redimiento del análsis sentimental.

Por ejemplo, teniendo un clasificador con dos posibles clases (*positiva* y *negativa*), para la etiqueta correspondiente a que la clasificación es *«positiva»* (etiqueta verdadera), se contabiliza el número de instancias fueron clasificadas como *«positivas»* (la predicción fue correcta), y después cuantas fueron clasificadas como *«negativas»* (clasificaciones incorrectas). Como se tienen dos posibles etiquetas, la matriz tendrá una dimensión de dos columnas por dos renglones sin incluir sus correspondientes encabezados. En la tabla 3.1 se ilustra lo descrito anteriormente, así como los valores que corresponden a cada celda de la matriz de confusión.

		Predicción	
		Positivo	Negativo
Real	Positivo	Verdaderos Positivos (VP): cantidad de positivos que fueron clasificados correctamente como positivos por el modelo.	Falsos Negativos (FN): cantidad de positivos que fueron clasificados incorrectamente como negativos.
	Negativos	Falsos Positivos (FP): cantidad de negativos que fueron clasificados incorrectamente como positivos.	Verdaderos Negativos (VN): cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.

Tabla 3.1: Matriz de confusión.

La matriz de confusión informa de la cantidad de clasificaciones con falsos positivos, falsos negativos, verdaderos positivos y verdaderos negativos. Permite un análisis más detallado que la proporción de aciertos correctos (precisión). También a través de esta matriz se puede evaluar la calidad del modelo en detalle, aplicando una métrica estándar como especificidad, precisión, etc.

3.2.2 Técnica de agrupamiento con lógica difusa

Esta técnica emplea algoritmos de agrupamiento, es una de las técnicas frecuentemente utilizada en el reconocimiento de patrones. En la literatura comúnmente se encuentra como *Fuzzy C-Means* y cabe mencionar que esta técnica basa su funcionamiento en el método *K-means*, difiriendo del método original en el hecho de un que vector (un registro o tupla, tweet en este caso) prescinde de la pertenencia exclusiva a un único grupo.

Por lo anterior, *Fuzzy C-Means* al igual que *K-Means* requiere un centro inicial para cada uno de los grupos (generalmente se elige de manera aleatoria), a partir de este centro durante cada iteración del algoritmo se emplea una métrica que permite definir qué miembros pertenecen a qué grupo, siendo la métrica euclidiana la más utilizada. A partir de los elementos que conforman al grupo, se calcula un nuevo centro para cada grupo, este puede ser diferente al grupo actual o no, convirtiéndose este posible cambio de centros como un criterio de terminación de la ejecución algoritmo, es decir, cuando ya no existen más cambios en los centros de los grupos se finaliza la ejecución. Sin embargo, este no es el único criterio de parada que se puede emplear, también se suele recurrir al número de iteraciones máximas para que se ejecute el algoritmo.

Para determinar el grado de pertenencia de un vector a cada grupo se calcula la similitud entre dicho vector y un grupo mediante una función, llamada función de pertenencia, la cual toma valores entre cero y uno. Los valores cercanos a uno indican una mayor similitud, mientras que los cercanos a cero indican una menor similitud. Por lo tanto, el problema del agrupamiento difuso se reduce a encontrar una caracterización de este tipo que sea óptima.

3.2.2.1 Transformación mediante técnica TF-IDF

Debido a que diversos algoritmos basan su funcionamiento en una métrica numérica, tal como acontece con el método de agrupamiento con lógica difusa, surge la necesidad de obtener una representación numérica de los datos cuando estos carecen de ella. En este trabajo se trabajarán con datos textuales, por lo que es necesario representar los datos de forma numérica, para lo cual TF-IDF (del inglés Term frequency – Inverse document frequency) es un medida que permite obtener dicha representación numérica.

De acuerdo con Ramos[17], esta es una técnica eficiente para la identificación de aquellos términos de mayor relevancia sobre una colección de textos. De esta forma, a partir de este método se obtiene una representación numérica de cada tweet a partir de los términos que conforman el corpus.

Mediante TF-IDF es posible obtener la representación numérica necesaria. Esta es una medida numérica que expresa cuán relevante es una palabra para un documento (tweet en este caso) en una colección. Por un lado el TF (Term Frecuency) mide la frecuencia de uso de un término específico en una página o documento. Cuanto más largo sea un contenido mayor serán las veces que se use una keyword dentro del mismo. Mientras que el IDF (Inverse Document Frecuency) mide la importancia de un término específico por su relevancia dentro del documento.

3.2.2.2 **Evaluación de algoritmos de agrupamiento: índices de calidad**

Es de importancia evaluar el resultado de los algoritmos de agrupamiento; sin embargo, es difícil definir cuando el resultado de un agrupamiento es aceptable. Por esta razón existen índices para la validación de los resultados de un agrupamiento.

Existen dos tipos de validación, el primero de ellos validación externa y la validación interna[18]. La principal diferencia es si se usa o no información externa para la validación, es decir, información que no es producto de la técnica de agrupación utilizada.

A diferencia de técnicas de validación externas, las de validación interna miden el algoritmo de agrupamiento únicamente tomando en cuenta la información de los datos, es decir, se evalúa que tan buena es la estructura de los grupos formados sin necesidad de información adicional.

Las métricas de validación interna pueden usarse para escoger el mejor algoritmo de agrupamiento, así como el número de clúster óptimo sin ningún tipo de información adicional, sin embargo, comúnmente se carece de este tipo de información.

Dentro de las métricas internas se encuentra el índice *Davies-Bouldin*, en el cual cuando se obtiene un índice pequeño indica que los grupos son compactos, e implica que los centros están bien separados unos de los otros. Esta técnica analiza cada grupo, uno por uno.

Para cada grupo se determina aquella relación máxima existente entre el la suma de la distancia promedio del centro del grupo a evaluar con respecto a sus vectores más la distancia del centro que se está evaluando con respecto al centro de otro grupo divido entre la distancia de los centros de

ambos grupos. En consecuencia, el número de grupos que minimiza esta métrica se toma como el óptimo.

3.2.3 Método *El Codo*

Uno de los problemas que se deben de atender cuando se trabaja con algoritmos de agrupamiento es la determinación del número óptimo de grupos con los que se debe de trabajar, para lo cual existen diversas técnicas, entre las cuales una de las más utilizadas es el método *elbow* o *el codo*[19], el cual utiliza los valores de la inercia obtenidos al ejecutar el algoritmo de agrupamiento para un número diferente de grupos (desde 2 hasta N), siendo la inercia la suma de las distancias euclidianas al cuadrado de cada vector del grupo al centro del mismo. De forma matemática se expresa de la siguiente forma:

$$Inercia = \frac{1}{N} \sum_{i=0}^{N} || x_i - \mu ||^2$$

La representación gráfica lineal de la inercia respecto del número de grupos permite apreciar un cambio brusco en la evolución de la inercia, siendo que la curva representada tiene una forma similar a la de un brazo y su codo. El punto en el que se observa ese cambio brusco en la inercia indicará el número óptimo de grupos a seleccionar. Es posible continuar con el análisis hasta encontrar un segundo cambio brusco en los valores calculados de la inercia. Sin embargo, al recurrir a esto lo que realmente se hace es subagrupar los grupos que se formaban al ejecutar el algoritmo de agrupamiento con el número de grupos identificado en el primer cambio brusco en la curva.

Cuando se trabaja con el número de grupos determinado a través de este método, al visualizar la distribución de los vectores en el espacio, se aprecia que el método del codo devuelve unos valores muy coherentes y se ajusta a los resultados esperados.

3.3 Características de las fuentes de la información

La cantidad de información que se genera a través de los diversos servicios que existen en la actualidad se caracterizan principalmente por tener un crecimiento exponencial.

Por ejemplo, la bitácora de un servidor web que atiende las peticiones de sus clientes, inicialmente el servidor atiende cierto número pequeño de peticiones, pero en medida de que el negocio va

creciendo, el número de peticiones que debe de atender el servidor también lo hará, lo cual se verá reflejado en el número de líneas que serán generadas en la bitácora.

El número de peticiones atendidas en un intervalo de tiempo aumentará, siendo que en consecuencia la velocidad con la que el servidor web responderá también deberá de aumentar, de lo contrario se comenzarán a presentar fallas en el servicio.

Analizar manualmente grandes cantidades de datos producidas en periodos de tiempo tan pequeños (horas, minutos o inclusive segundos) resulta no ser una opción viable.

Esto debido a diversas causas, en primer lugar, se presentarán dificultades para manipular grandes volúmenes de datos. En el ejemplo del servidor web, los servidores son equipos de cómputo con características especiales que cuentan con los servicios necesarios para atender tareas tan demandantes (en aspectos de memoria, capacidad de procesamiento, etcétera), por lo cual exportar toda la bitácora de un servidor a un equipo de cómputo que no cuente con características similares será un desperdicio de recursos ya que dicho equipo de cómputo carecerá de la capacidad para ejecutar los procesos analíticos deseados.

Además de las dificultades para el almacenamiento de grandes volúmenes de datos se identifica otra dificultad al tratar de analizarla, debido a que la información puede provenir de fuentes diversas, implica que estas no tendrán necesariamente el mismo tipo de dato, conllevando a la tarea de manipular datos estructurados, semi-estructurados y/o no estructurados e integrarlos para el proceso de minado de datos.

3.4 Big Data

3.4.1 Características de Big Data

Las dificultades que se han mencionado en el capítulo anterior se pueden encontrar en diversos sistemas de información de diferentes ámbitos, tales como empresas, redes sociales, universidades, videojuegos e *Internet de las Cosas*. Estos son solo algunos ejemplos de áreas que dan lugar a la concepción de Big Data.

Es importante destacar el hecho de que en la actualidad no hay una definición generalizada del término Big Data. Sin embargo, Laney Doug, pionero en el área, ha presentado la siguiente definición: *"«Big Data» son activos de información de gran volumen, velocidad y variedad que exigen formas rentables e innovadoras de procesamiento de la información para mejorar el conocimiento y la toma de decisiones"*[20].

Las tres características que se han mencionado sobre Big Data son justamente aquellas que singularizan el término y suelen ser denominadas las *3 V's de Big Data*, aunque hay autores que difieren y suelen añadir otras características como son: el *valor*, la *volatilidad*, *veracidad* y *validez* de la información.

Big Data se encuentra presente en nuestras actividades cotidianas, desde las redes sociales y multimedia, pasando por las transacciones que generamos al realizar nuestras compras diarias, redes de sensores, hasta los dispositivos de generación de contenido, el tamaño de la información va en constante crecimiento, lo cual en consecuencia demanda una mayor capacidad de procesamiento y de almacenamiento. Cuando se desea realizar un análisis de este tipo de datos se llega a la problemática de trabajar con datos que pueden ser estructurados, semi o no estructurados y originalmente las herramientas analíticas trabajaban únicamente con datos estructurados. Lo cual conlleva a que se tiene que optar por nuevas alternativas de almacenamiento y es conveniente resaltar que esto se torna en una tarea compleja.

Por ejemplo, el análisis de tweets presenta ciertas las dificultades, por ejemplo, la información es producto de diferentes culturas (costumbres, tradiciones, estructuras políticas y religiosas, idiomas, etcétera). Durante la adquisición de los datos se requieren técnicas de limpieza e integración antes de ser almacenados, y posteriormente la organización y el almacenamiento de los datos debe ser adecuado (puede ser un HDFS o en la nube), y se deben de tomar en consideración requerimientos de privacidad y seguridad en la información[21].

3.4.2 Enfoques de Big Data

De acuerdo con Chandra[13], existen tres enfoques que suelen dar solución a aplicaciones para análisis de Big Data, sin embargo estos no son los únicos que permiten hacer frente a una problemática de esta naturaleza, existen otras aproximaciones, pero las que se presentan a continuación suelen ser frecuentemente recurridas debido a la estabilidad y las herramientas que ofrecen.

Estos enfoques son:

a) MapReduce (Hadoop).
b) BSP (HAMA).
c) STORM.

Enfoque MapReduce

MapReduce (Hadoop) es un modelo de programación de Google propuesto en 2004; de acuerdo con Jeffrey Dean[22], MapReduce se define como un modelo de programación en paralelo para el procesamiento y análisis distribuido de grandes volúmenes de datos.

A través de este modelo es posible especificar una función de mapeo (*map*), la cual procesa una pareja de tipo *«llave, valor»*, en donde el valor corresponde al término a buscar y la llave es generada por el algoritmo; con este par se genera un conjunto intermedio de parejas de tipo *«llave, valor»* con todas las ocurrencias que se encuentren del parámetro de búsqueda sobre todos los nodos de información, y partir de este conjunto intermedio la función de reducción (*reduce*) se encarga de realizar una unificación de todos los valores intermedios asociados a la misma *llave* intermedia. En la figura 3.1 se presenta un diagrama en el cual se muestra el procesamiento MapReduce.

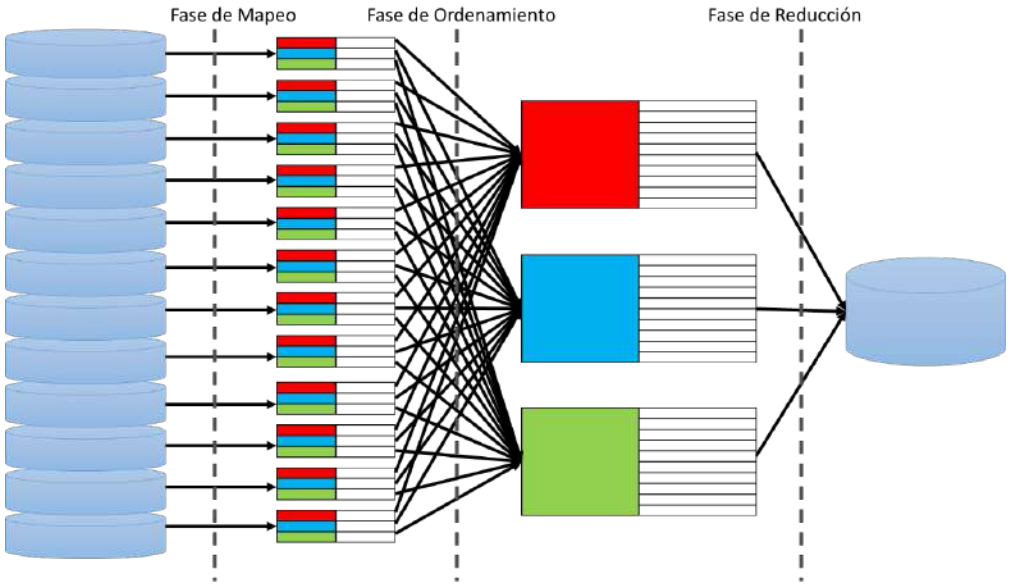


Figura 3.1: Esquema de funcionamiento basado en MapReduce.

En la figura 3.1 del lado izquierdo encontramos la distribución del sistema de almacenamiento, sobre los cuales se realiza la búsqueda del término deseado a través de la generación de la pareja *«llave, valor»*, la cual, para la búsqueda de un término en específico será de la forma *«llave, término-a-buscar»*, en este ejemplo se tienen tres términos a buscar (cada uno correspondiente a un color). En la *fase de mapeo* todas las instancias que sean encontradas a través de todos los discos son regresadas para la *fase de ordenamiento*, en esta fase se juntan todas las instancias que fueron encontradas para cada una de las llaves, la salida de esta fase es enviada a la *fase de reducción* en la cual se eliminan resultados duplicados para que las instancias únicas sean devueltas como resultado de la búsqueda.

El motor de búsqueda de Google utiliza este modelo, el término que se desea buscar genera un par de tipo *«llave, término-a-buscar»*, y el motor distribuye la búsqueda de la pareja a través de los diferentes nodos, generando conjuntos intermedios donde tenga una coincidencia con el criterio de búsqueda. Se realiza un ordenamiento de tales conjuntos, posteriormente se realiza la eliminación de registros duplicados y el resultado de dicha unificación es lo que nos devuelve como resultado de la búsqueda. El sistema ofrece la capacidad de distribuir la búsqueda de forma paralela.

Enfoque BSP

BSP (*Bulk Synchronous Parallel*) es un modelo de cómputo que trabaja con el framework Apache HAMA el cual fue liberado en 2012 y creado tomando como inspiración los sistemas Pregel[1] y DistBelief[2] de Google. BSP al igual que Map Reduce es un modelo de cómputo paralelo y distribuido.

Un sistema basado en BSP *"permite trabajar con datos a gran escala (especialmente datos gráficos")*[25]. El sistema tiene la capacidad de configuración flexible y extensibilidad para funciones y estrategias (como el ajuste de los parámetros en función del volumen de datos) para procesar datos a gran escala, presenta gran tolerancia a fallos, además, a través de estas estrategias logra equilibrar la carga y ejecutar algoritmos de agrupamiento o clasificación en conjuntos de datos.

[1]Pregel es un framework de programación distribuida, enfocado en proporcionar a los usuarios una API para la programación de algoritmos de gráficos mientras se gestionan los detalles de la distribución de forma invisible, incluyendo la mensajería y la tolerancia a fallos[23].

[2]Es un framework que puede utilizar clusters de computación con miles de máquinas para entrenar modelos grandes[24].

Un sistema de tipo BSP consiste de tres componentes:

1. Componentes con capacidad de procesamiento y/o manejo de memoria.
2. Una red que permita la comunicación entre los distintos nodos que componen al sistema.
3. Hardware que permita la sincronización de todos los componentes del sistema.

Este modelo puede ser entendido como un conjunto de procesadores de entre los cuales, existe un nodo designado como *master*, el cuál será el encargado de coordinar a todos los demás nodos, a través de los cuales se almacenarán datos y se ejecutará el procesamiento de dichos datos.

El modelo BSP es un modelo de cómputo paralelo basado en *super-step*, y en cada uno de estos *super-step* se lleva a cabo la comunicación entre los nodos, logrando así la coordinación y sincronización del trabajo. En el caso de que exista dependencia entre nodos, estos se envían mensajes para solicitar los recursos que requieran para completar su tarea. Este proceso se ilustra en la figura 3.2.

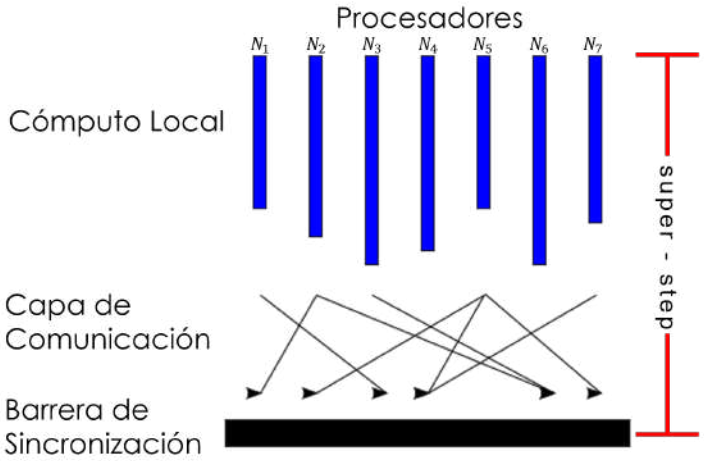


Figura 3.2: Arquitectura y funcionamiento de BSP. (Extraída y traducida de [26]).

El siguiente *super-step* puede comenzar hasta que el cálculo de cada procesador ha finalizado y se han completado los mensajes de envío y recepción de cada tarea.

En la figura 3.2 se observan siete procesadores $(N_1, N_2, ..., N_7)$, cada uno de estos componentes está encargado de realizar el cómputo local que se le ha designado por el nodo *master*, a través de la *capa de comunicación* cada nodo envía los correspondientes resultados del cómputo que

ha realizado, sin embargo, gracias a la *barrera de sincronización* ningún nodo puede continuar realizando el proceso de cómputo hasta que terminen todos los nodos de realizar sus cómputos locales actuales y envíen sus respectivos resultados.

Hasta que todos los nodos han enviado respuesta, es entonces cuando se inicia el siguiente *super-step*, continuando así con el procesamiento de *cómputo local* en cada procesador. Siendo que el nodo *master* será el encargado de brindar las respuestas de manera global en el sistema.

Lo que diferencia principalmente a BSP de MapReduce es el hecho de que los algoritmos implementados en BSP requieren que los datos sean almacenados y procesados en múltiples *super-step*, lo cual no puede ser implementado en MapReduce si el número de *super-step* no es constante, esto se debe a la naturaleza del framework, pues no permite el almacenamiento hasta alcanzar la barrera de sincronización, posicionando de este modo a MapReduce como una alternativa más eficiente para el procesamiento masivo de datos.

Enfoque STORM

Storm es un proyecto de Apache de código abierto, que de acuerdo con el trabajo propuesto por Ankit Toshniwal[27], lo define como *"un motor de procesamiento de datos de streaming distribuido en Twitter"*, que potencia las tareas de gestión de datos en streaming que son cruciales para proporcionar servicios de Twitter.

Apache Storm es una tecnología que actualmente es usada por un gran número de compañías tales como Yahoo!, Twitter o Flipboard. Storm permite llevar a cabo análisis en tiempo real.

Un sistema que emplea Storm está definido por una topología que describe la forma en la que operan sus componentes, estos son de dos tipos: *spout* y *bolts*. Esta topología se presenta en el diagrama que se muestra en la figura 3.3.

Los *spout* son los encargados de capturar el streaming, típicamente tienen como entradas registros de fuentes de datos externas (por ejemplo, la API de Twitter). Los *bolts* procesan los datos que son proporcionados por los *spout* y los encapsulan para que pueda ser procesados por otros nodos.

La arquitectura de Storm es bastante sencilla, se divide en los siguientes componentes: *Nimbus*, *ZooKeeper* y *Supervisor*. En el diagrama de la figura 3.4 se aprecia la organización de dichos componentes.

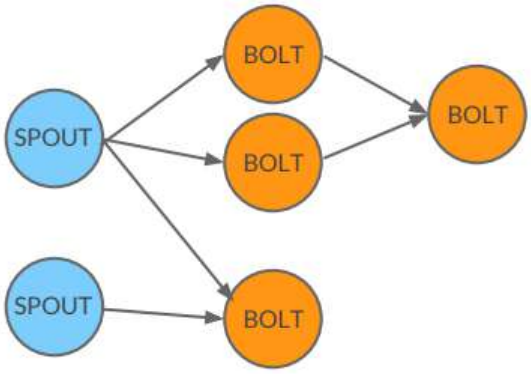


Figura 3.3: Topología de Apache Storm. (Obtenida de [28])

En donde el *nimbus* se encarga de distribuir el código entre los demás nodos, y también se encarga de las asignaciones de las tareas y monitorea los fallos. El *zookeeper* conserva la información de las máquinas que están ejecutando las tareas de procesamiento (para propósitos de coordinación). Finalmente, el *supervisor* escucha a cada uno de los procesadores que le fueron asignados, comienza y termina los procesos a través de comandos en *Nimbus*.

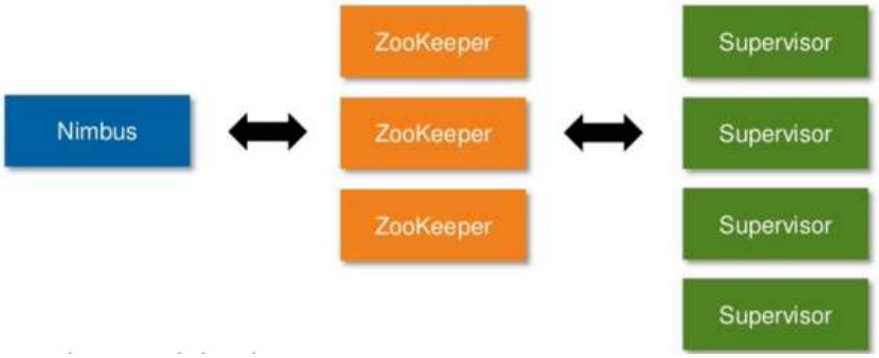


Figura 3.4: Detalle de la arquitectura de Apache Storm. (Obtenida de [28])

Storm se caracteriza por ser escalable, flexible, extensible, eficiente y fácil de administrar, además este motor se ejecuta sobre un conjunto de nodos, entre ellos existe uno denominado como *master* y este es el responsable de distribuir y coordinar la ejecución de las consultas. Cada nodo contiene uno o más *procesadores* y estos son los encargados de realizar las operaciones que han sido asignadas al nodo, y los resultados que se obtengan a partir de dicha ejecución serán devueltos al nodo *master*.

Lo que diferencia a *Storm* de otras soluciones de Big Data es el paradigma que aborda. Hadoop es fundamentalmente un sistema de procesamiento por lotes. Los datos se introducen en el sistema de archivos de Hadoop (HDFS) y se distribuyen a través de los nodos para su procesamiento. Cuando el procesamiento está completo, los datos resultantes regresan a HDFS para ser utilizados por el programador. *Storm* soporta la construcción de topologías que transforman secuencias de datos sin finalizar. Esas transformaciones, a diferencia de los trabajos de Hadoop, nunca se detienen, sino que continúan procesando datos conforme van llegando.

3.4.3 **Tipos de soluciones**

Tal como acontece con las soluciones de software que utilizamos en nuestras actividades cotidianas, encontramos dos vertientes para poder adquirir un software. Por un lado, contamos con aquellas distribuciones que son libres, las cuales no requieren que sea cubierto costo alguno. Sin embargo, este tipo de soluciones conlleva el inconveniente de que no se garantiza el óptimo funcionamiento del software, ni tampoco se cuenta con alguna empresa u organización que respalde ni brinde soporte al producto. Por el otro lado se encuentran los productos comerciales, los cuales garantizan el funcionamiento del software, o su respectivo soporte en caso de fallas, así como mantenimiento para este mismo a cambio de un pago ya sea periódico o único.

Soluciones comerciales

Existen diversas soluciones comerciales, estas son usadas principalmente por grandes corporativos debido a que son los únicos grupos que cuentan con la capacidad de solventar sus costos. Se identifican a los tres proveedores principales en la industria que son:

- SAP HANA.
- Oracle.
- Terada.

SAP HANA

SAP HANA y *SAP Cloud Platform Big Data Services* es proveedor libre de Big Data como servicio. Esta plataforma opera como una refinería de datos, en la cual se limpian y adecúan los datos de manera masiva antes de que estos sean analizados a través de SAP HANA.

La infraestructura informática y de almacenamiento de datos masivos de Big Data Services (BDS) se encuentra basado en Hadoop, lo cual hace que BDS sea una plataforma ideal para almacenar y procesar grandes volúmenes de datos, ya sea que estos tengan o no una estructura, teniendo como ventaja adicional la baja latencia en la realización del análisis.[29]

Un beneficio adicional que se obtiene al manejar SAP HANA conjuntamente con SAP Cloud Platform Big Data Services es la opción de reubicar datos más antiguos (o de menor valor) en sus entornos Hadoop. Los grandes servicios de datos están bien adaptados para almacenar grandes volúmenes de datos a los que se accede con poca frecuencia, a la vez que se cuenta con la posibilidad de consultar dichos datos, cuando sea necesario, a través de las funciones de virtualización de datos de SAP HANA, denominadas acceso inteligente a datos de SAP HANA. En la figura 3.5 se ilustra la arquitectura BDS en conjunto con SAP HANA y su integración con aplicaciones empresariales.

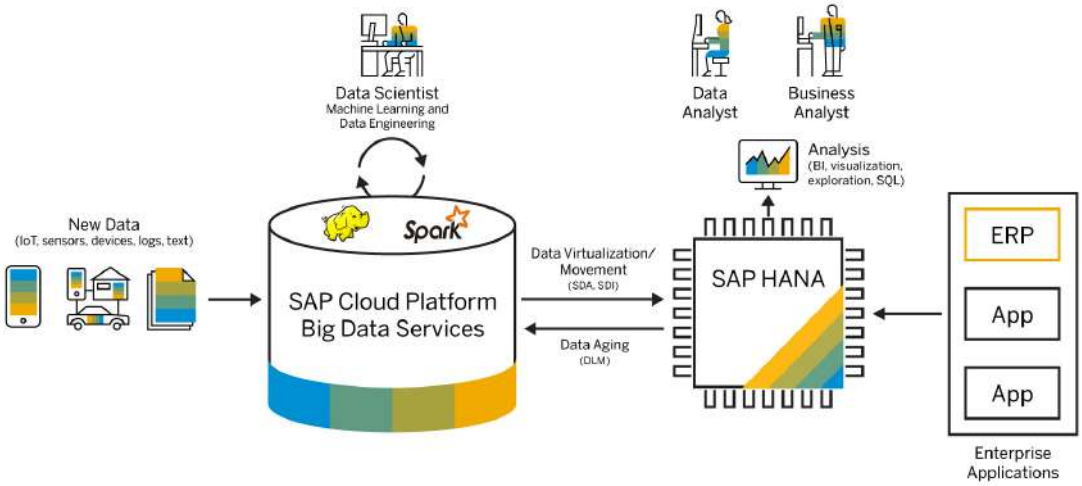


Figura 3.5: Arquitectura de SAP Cloud Plataform Big Data Services. (recuperada de [29])

La conectividad entre SAP HANA y SAP Cloud Platform Big Data Services se logra mediante el uso de la integración de datos inteligentes de SAP HANA (SDI) y el acceso inteligente a datos (SDA). SDI permite el movimiento de datos entre Big Data Services y SAP HANA. Mientras que SDI permite traer subconjuntos más pequeños de datos preprocesados a SAP HANA.

Un ejemplo en donde SAP HANA se utilizó para un procesamiento analítico de grandes cantidades de datos fue en la copa mundial de fútbol 2014, la federación alemana de fútbol a través de SAP HANA logró realizar un análisis de una cantidad masiva de vídeos sobre las jugadas de los miembros del equipo, a través de dicho análisis se logró extraer información de métricas sobre cada jugador[30].

Oracle

Oracle se distingue por contar con una gran variedad de productos, adecuando cada uno de ellos a una tarea particular. Lo cual implica que entre más productos necesitemos, el costo por mantener la arquitectura será mayor.

Entre los productos que Oracle ofrece se encuentran *Data Integration Platform*, el cual brinda la posibilidad de realizar integración de los datos, así como calidad y gobierno de datos, además esta herramienta incorpora algoritmos de Aprendizaje de Máquina e Inteligencia Artificial[31]. Por otro lado, *Oracle Event Hub Cloud Service*, brinda el potencial de Apache Kafka como una plataforma administrada de transmisión de datos integrada en el ecosistema de Oracle Cloud[32]. Kafka es una plataforma de transmisión de datos gestionada integrada en el ecosistema Oracle Cloud. Oracle permite obtener información basada en datos y ejecute acciones al conectar, analizar e integrar los datos de dispositivos mediante *Internet of Things de Oracle*[33] y *Oracle Big Data Cloud*[34] cuenta con Apache Hadoop y Apache Spark se ofrecen como una plataforma administrada, escalable e integrada para el análisis masivo de datos. Además de contar con una gama aplicaciones analíticas para la toma de decisiones[35].

Teradata

De manera análoga a Oracle, Teradata ofrece una gran variedad de productos especializados en cada una de las tareas particulares para el manejo de *Big Data*, tal como se aprecia en la figura 3.6, se observan productos como *Data Warehouse*, *No SQL*, *In Memory*, etcétera.

Teradata comenzó a comercializar con productos de tipo Big Data en 2010. El aumento de los datos semi-estructurados y no estructurados recopilados a partir de las nuevas fuentes de información, tales como las redes sociales, condujo a Teradata a formar el *Petabyte Club* en 2011 para sus grandes usuarios con grandes cantidades de datos[37].

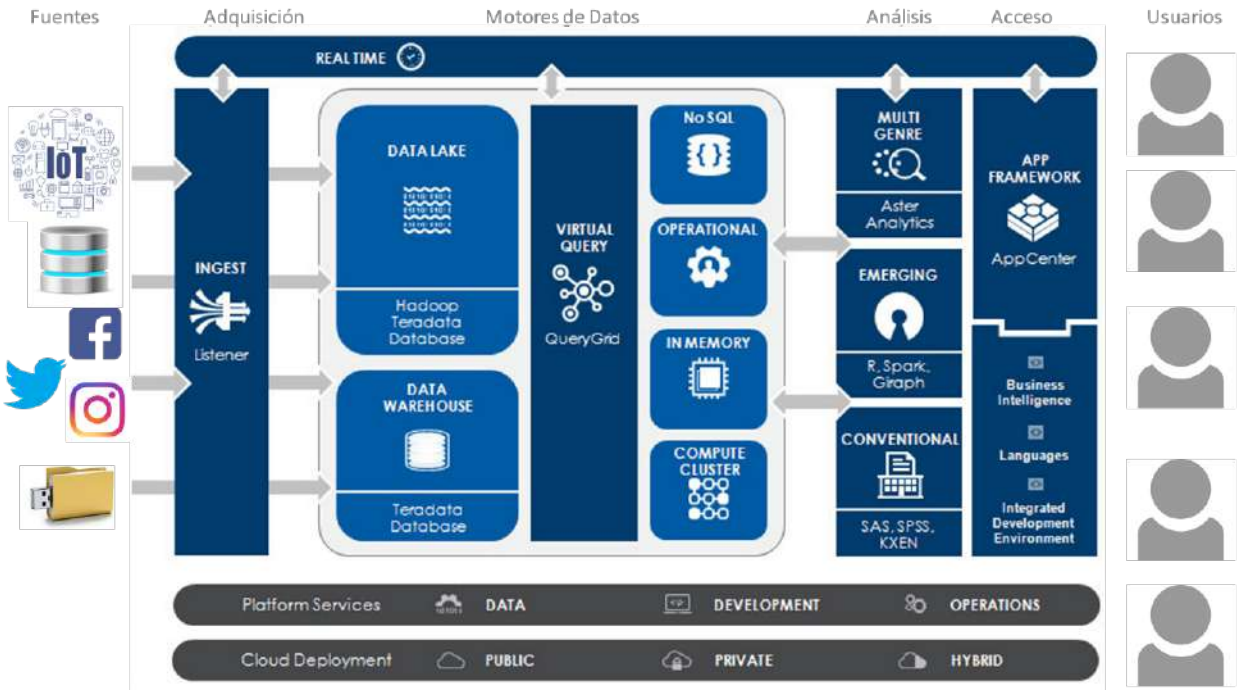


Figura 3.6: Arquitectura de aplicaciones para Big Data de Tearadata. (extraída de documentación de Teradata[36])

Para Teradata, Big Data impulsó la adquisición de Aster Data Systems en 2011, a través del cual comenzó a brindar la operación del modelo MapReduce y la capacidad de almacenar y analizar datos semi-estructurados[38].

Soluciones de distribución libre

Se encuentran disponibles opciones de distribución libre, con la ventaja de que no se requiere realizar algún pago para la manipulación del software. Se debe de tener en cuenta la existente posibilidad de fallos y el hecho de no contar con soporte de manera inmediata. Una de las opciones utilizadas con mayor frecuencia debido a las ventajas que ofrece es Hadoop.

Hadoop

El sistema de archivos distribuidos HDFS trabaja con el framework Hadoop, dado que es de libre distribución suele ser la solución más utilizada por muchas organizaciones que realizan trabajos de ambiente Big Data.

Hadoop está desarrollado como una plataforma de procesamiento de datos distribuida para el análisis de grandes cantidades de datos. Las empresas pueden analizar grandes volúmenes de datos conteniendo la información sensible de los usuarios mediante el uso de Hadoop y utilizarlos para su comercialización[39].

El modelo de cómputo distribuido de Hadoop procesa Big Data a gran velocidad. Cuantos más nodos de cómputo sean utilizados, mayor poder de procesamiento se tendrá. Además, el procesamiento de datos y aplicaciones está protegido contra fallos del hardware. Si falla un nodo, los trabajos son redirigidos automáticamente a otros modos para asegurarse de que no falle el procesamiento distribuido. Se almacenan múltiples copias de todos los datos de manera automática.

En la figura 3.7 se aprecia la arquitectura de Hadoop, la cual cuenta con diversas extensiones que en conjunto permiten implementar aplicaciones para ambiente Big Data.

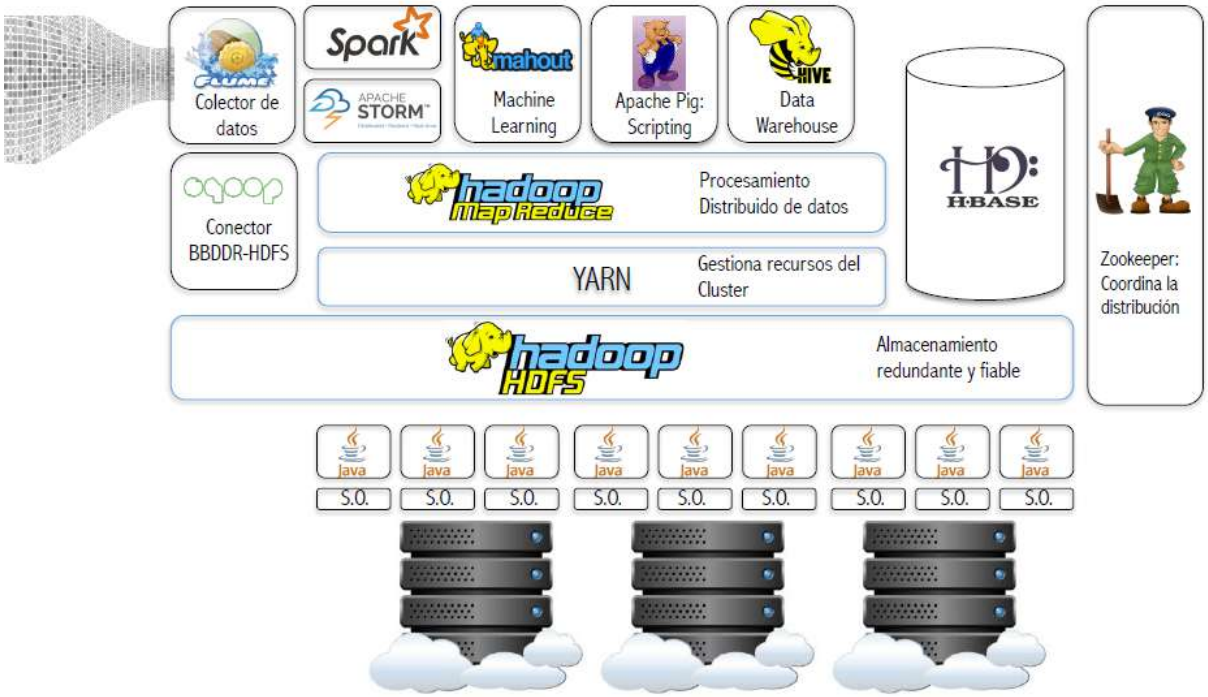


Figura 3.7: Arquitectura del framework Hadoop. (extraída de Holmes[40])

La arquitectura del framework Hadoop cuenta con una capa de almacenamiento, la cual está conformada por grupos de discos, ya sean locales o en la nube (parte inferior de la figura 3.7), sobre estos se tiene el sistema de archivos HDFS quien será el encargado de la gestión del almacenamiento y recuperación de los datos; a través de la capa *YARN* se logran gestionar los recursos de los

nodos, YARN es una reescritura de software que desacopla las capacidades de gestión de recursos y planificación de la capa MapReduce, permitiendo a Hadoop soportar enfoques más variados de procesamiento, y una gama más amplia de aplicaciones. Por ejemplo, los clusters Hadoop ahora pueden ejecutar consultas interactivas y transmisiones de aplicaciones de datos de forma simultánea con los trabajos por lotes de MapReduce.

Para llevar a cabo la coordinación de todo este trabajo se encuentra *Zookeeper*, quien ofrece un servicio para la coordinación de procesos distribuidos y altamente confiable que da soluciones a varios problemas de coordinación para grandes sistemas distribuidos.

Adicionalmente en la última capa se encuentran frameworks como *Spark* (para realizar cómputo sobre clústers), *Storm* (captura y análisis de datos en tiempo real), *Apache Pig* (permite la creación de programas de tipo MapReduce en un lenguaje de alto nivel), *Apache Hive* (estructura de almacenamiento y consulta de datos sobre Hadoop), etcétera, todos estos ofrecen funcionalidades diversas que brindan en conjunto mayor potencial a Hadoop para la realización de procesos analíticos.

3.5 Conclusiones del capítulo

Después de lo expuesto en este capítulo cabe resaltar que Big Data se encuentra presente en nuestras actividades cotidianas a pesar de que no sea perceptible a simple vista. Con estas actividades generamos una cantidad de datos de diversos tipos, y tratar de realizar un análisis de dichos datos a través de técnicas convencionales ya no es viable. Por lo cual se requiere optar por soluciones tecnológicas con las características necesarias para un proceso analítico de tipo Big Data. En consecuencia se puede decir que Big Data permite dar solución a problemas reales que no pueden ser atendidos mediante herramientas convencionales.

La importancia de este proceso analítico radica en el hecho de que a partir de este se puede obtener información de utilidad para el negocio. Tal como se mencionó en este capítulo, a partir de un análisis sobre Big Data es posible determinar un próximo cliente, saber si una campaña publicitaria tendrá éxito, identificar tendencias, etc.

Sin embargo, es importante resaltar que no todos los problemas son de tipo Big Data, antes de emitir un juicio se deben de identificar las características del problema que se desea atender, en caso de que se cumpla con las que han sido descritas en este trabajo (las 3V's Big Data) entonces

es hasta este punto cuando se puede considerar optar por una solución de tipo Big Data.

Por otro lado, esta tecnología es reciente en comparación con otras soluciones tecnológicas, lo cual implica que sigue en desarrollo, adecuándose a las problemáticas que se desean resolver. Sin embargo, hoy en día contamos con herramientas que permiten dar solución a gran parte de los problemas a los que se enfrenta la Ciencia de Datos.

Finalmente, cabe destacar que las grandes colecciones de datos conducen continuamente a innovaciones en todos los sectores, tales como la industria, universidades, salud, gobierno, economía, entre otros. Siendo aquí justamente donde Big Data toma presencia ya que permite analizar dichas colecciones de datos.

De acuerdo con la fuente de información identificada (Twitter) a través de la problemática que se pretende resolver, se identifican las características de dicha fuente de información. En primer lugar se encuentra el hecho de que la información a manipular es no estructurada, es decir, carece de una estructura explícita, lo cual implica que no se puede emplear una herramienta analítica convencional. Por otro lado, teniendo como limitante aspectos de hardware y software en cuanto a la capacidad de almacenamiento y procesamiento no se podrán manejar grandes volúmenes de datos. Sin embargo, con el trabajo que se realice en futuras secciones se podrá definir un marco de trabajo, el cual podrá ser escalado a hardware que cuente con los requerimientos necesarios para manejar grandes cantidades de datos. En consecuencia, para el manejo de cantidades masivas de datos se identifica una demanda de respuesta para el almacenamiento en tiempos de lapsos muy cortos, de lo contrario se podría perder el flujo de información.

A partir de lo anterior, se considera que de acuerdo con la naturaleza de la fuente de información es necesario implementar el proceso analítico en un ambiente Big Data, ya que las características de la fuente de información impiden que dicho proceso analítico sea realizado mediante herramientas convencionales.

Por último, a partir de la identificación de los tipos de soluciones tecnológicas existentes para la implementación de una aplicación analítica se deduce que las soluciones de distribución libre son la opción más viable para este trabajo, ya que con este tipo de soluciones se cuenta con la tecnología necesaria para implementar la aplicación analítica de interés con la ventaja de que no se requiere cubrir ningún tipo de soporte o derecho para hacer uso de estas herramientas.

Suppose that we are wise enough to learn and know - and yet not wise enough to control our learning and knowledge, so that we use it to destroy ourselves? Even if that is so, knowledge remains better than ignorance.

Isaac Asimov

CAPÍTULO

4

Metodologías de desarrollo de aplicaciones de tipo Big Data

Actualmente acontece que en lo concerniente al desarrollo de aplicaciones que trabajan con diferentes tipos de datos, entre ellas las apliaciones analíticas Big Data, por tratarse de un fenónomeno relativamente nuevo aún se carece de estándares o metodologías precisas y detalladas para dicho tipo de sistemas, sin embargo, esto no impide tener la concepción de una metodología para el desarrollo de este tipo de aplicaciones. En la actualidad se cuenta con diversos trabajos cuya temática se centra en Big Data, presentando una problemática en particular en cada uno de estos, y para cada problemática se propone una solución mediante el uso de una herramienta de tipo Big Data. A lo largo de cada uno de estos trabajos se detallan las fases que lo conformaron, abordando las tareas y retos que tuvieron que ser atendidos para poder llegar a la solución del problema de interés de cada documento. Esto con el objeto de identificar los pasos en común que apliquen para la problemática de este trabajo.

4.1 La analítica de Big Data en el sector de salud: promesa y potencial

4.1.1 Introducción

Raghupathi et al., a través de su artículo *Big data analytics in healthcare: promise and potential*[41] brinda una perspectiva sobre cómo históricamente el sector de salud siempre ha generado grandes cantidades de datos, esto derivado de los requisitos regulatorios con el paciente así como del seguimiento de su historia clínica. Almacenar y procesar volúmenes de datos se torna una tarea que no puede ser atendida con procesos convencionales.

4.1.2 Metodología para el desarrollo

Raghupathi presenta diversas etapas que en conjunto buscan mostrar el potencial que brinda el análisis de Big Data en este sector. Dichas etapas se presentan a continuación por orden de realización:

- *Caracterización de la situación actual*, siendo aquí donde se perciben las características del contexto, en este caso, se logra establecer la necesidad de llevar a cabo un proyecto analítico de tipo Big Data en el sector de salud.
- Definición de un *propósito* para llevar a cabo dicho proyecto, a través de esta etapa Wullianallur logra establecer cómo manejar la problemática en cuestión, identifica la importancia e interés del proyecto y finalmente, logra identificar los recursos con los que cuenta.
- La *adquisición* de los datos, esto se lleva a cabo a través de la unificación de las diversas fuentes de información, estas son internas y externas.
- *Transformación* es la etapa donde Raghupathi emplea diversas herramientas, tales como ETL[1], Data Warahouse (DW) y conversiones tradicionales desde formatos CSV[2] y tablas almacenadas en DBMS; la elección de las herramientas dependerá de las fuentes de información. Esta etapa se auxilia de plataformas y herramientas Big Data, tales como Hadoop,

[1]Es el proceso es facilitar el movimiento de los datos y la transformación de los mismos, integrando los distintos sistemas y fuentes en la organización.

[2]Son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

Pig, Hive, Jaql, Zookeeper, HBase, Cassandra, Oozie, Avro, Mahout y otros.

- Las *aplicaciones analíticas* corresponde a la última etapa, siendo aquí en donde se llevan a cabo los procesos analíticos, es decir, la ejecución de consultas, obtención de reportes, explotación de cubos OLAP y aplicación de técnicas de Minería de Datos con el propósito de diagnósticar a partir de los datos y mediante segmentaciones de los datos obtener un perfil de los pacientes.

4.1.3 Conclusiones

Como conclusiones de este trabajo se presenta que Big Data en el sector de salud está cobrando mayor relevancia a medida que la cantidad de datos va en aumento de manera significativa como resultado de las actividades cotidianas, siendo poco viable llevar a cabo un análisis de estos grandes grupos de datos mediante las herramientas analíticas convencionales, y esta es la principal razón por la cual es necesario recurrir a herramientas analíticas de tipo Big Data, la cuales cuentan con el potencial necesario para poder cumplir con el objetivo analítico.

Por otro lado, es importante destacar el hecho de que Wullianallur en este trabajo no identifica la visualización de los datos ni la toma de decisiones (diágnóstico a partir de datos y perfilado de pacientes) como etapas independiente, más bien, él incluye estos procesos como parte de las tareas involucradas dentro de la fase analítica.

4.2 Retos y oportunidades con Big Data

4.2.1 Introducción

Lanbrinidis y Jagadish en su trabajo *Challenges and Opportunities with Big Data*[9] buscan reconocer la importancia que el análisis de los datos en ambientes Big Data tiene en la actualidad, destacando el hecho de que actualmente no existe un consenso para la definición de término Big Data e inclusive se han presentado diversas discusiones controversiales al respecto.

Buscando contar con una formalización del proceso analítico para ambientes big data, en este trabajo se abordan las etapas necesarias para realizar un proceso analítico de esta naturaleza. Se

mencionan así mismo las herramientas tecnológicas empleadas para cumplir con su objetivo analítico.

4.2.2 Metodología para el desarrollo

Lanbrinidis y Jagadish a lo largo de su trabajo identifican las etapas que a continuación se enlistan, el orden de presentanción corresponde con el orden de implementación:

- Como primera etapa se identifica la *adquisición* de los datos, estos datos en crudo son almacenados en el sistema de información adecuado.
- Se considera importante obtener información respecto a la extracción de los datos (*metadatos*[1]), es decir, tener un repositorio sobre dónde se han obtenido los datos ya que esto podría ser de utilidad en especial para la identificación de información duplicada.
- Como tercera etapa que identifica Lanbrinidis se encuentra la *limpieza y adecuación* de los datos, siendo esto realizado a través de consultas de integración, limpieza y de acceso a los datos.
- Siendo en este punto viable proceder con el *análisis de información*, en esta etapa tiene lugar la ejecución de los algoritmos adecuados para alcanzar los propósitos. Tal como se mencionó en la introducción, Lanbrinidis y Jagadish buscan únicamente destacar los pasos de importancia para poder realizar un análisis Big Data, por lo cual no se menciona alguna técnica o algoritmo en particular.
- La *toma de decisiones* es la última etapa, y esta estará fundamentada a partir de los resultados reportes y otros documentos informativos obtenidos en la etapa anterior.

4.2.3 Conclusiones

Derivado de esta trabajo, Lanbrinidis y Jagadish ratifican la importancia del análisis de los grandes volúmenes de datos que se generan en la actualidad, y que para poder llevar a cabo un estudio analítico de este tipo se requieren aplicar las herramientas tecnológicas adecuadas.

[1]Los metadatos se caracterizan por ser datos altamente estructurados que describen características de los datos, como el contenido, calidad, información y otras circunstancias o atributos.

Además, el área de Big Data es un área prometedora y esto se respalda al revisar las inversiones multi millonarias realizadas por grandes industrias tecnológicas, tales como Amazon o Google.

4.3 Análisis sentimental de tweets

4.3.1 Introducción

En la actualidad muchas personas expresan su opinión sobre productos que utilizan en su vida cotidiana a través de micro blogs, haciendo que las empresas se interesen por este tipo de sitios, analizando cada entrada a estos micro blogs se encuentra la expresión de un sentimiento por un producto en particular. Agarwal, Xie, Vovsha, Rambow y Passoneau a través de su trabajo *Sentiment Analysis of Twitter Data*[42] realizan un análisis sentimental a un grupo de tweets, con el objetivo de obtener una suma de los sentimientos (polaridad[1]) de las publicaciones de los usuarios a través de Twitter.

4.3.2 Metodología para el desarrollo

Agarwal et al., identifican en su metodología empleada para el desarrollo las etapas que se presentan a continuación, el orden de presentación corresponden con el orden de implementación:

- La *recolección* de los datos es la primera etapa. Se recolectó un grupo de tweets, dicho repositorio se encuentra disponible para otros trabajos de investigación.
- En una primera aproximación se implementa un modelo de unigramas, el cual presenta un desempeño deficiente, por lo cual a partir de esto se opte por definir una fase de *preprocesamiento* a través de la cual emplean otros recursos, el primero de ellos es un diccionario que permite hacer el mapeo de los emoticonos hacia un sentimiento, (por ejemplo *:) :] :o* se pueden catalogar como algo positivo). Esta herramienta la refieren como características sentimentales (CS). Otra herramienta que emplearon es otro diccionario, pero este permite realizar un mapeo de las abreviaturas más comunes del idioma inglés, por ejemplo *gr8* o *gr8t* es mapeado como *great*. Y finalmente, la última herramienta que se emplea es un diccionario

[1] Asignando una de tres posibles etiquetas: positivo, negativo o neutral.

de palabras vacías, es decir, aquellas que no tienen un significado tales como son artículos, pronombres, preposiciones, etc.

- El *análisis* es la siguiente etapa, para la cual además de los modelos de unigramas y unigramas con preprocesamiento, se utilizó un Tree Kernel[1] donde se mapearon los tweets antes de haber pasado por la etapa de *preprocesamiento*. Después de obtener estos dos primeros resultados se optó por volver a ejecutar el modelo inicial (unigrama) pero con la diferencia de que ahora los datos han sido pre-procesados.

  Cómo se puede observar en la figura 4.1, se presentan la curvas de aprendizaje de los modelos utilizados, a partir de este gráfico se puede deducir que el *preprocesamiento* permitió la optimización de los modelos, pues logró mejorar el modelo un 5 % en exactitud, además de que los modelos que emplean el preprocesamiento a partir del 20 % de entrenamiento presentan valores de exactitud mayores al 70 %, lo cual se logra con el modelo original a partir de que se ha entrenado con el 90 % de los datos.

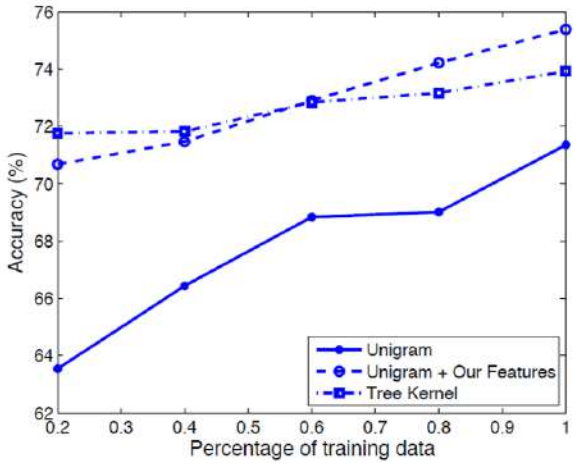


Figura 4.1: Curva de aprendizaje de los modelos empleados. Extraída de [42]

4.3.3 Conclusiones

El trabajo presentado por Agarwal et al., está muy bien estructurado, es muy robusto y ofrece un mayor grado de confiabilidad en el modelo obtenido a través del *preprocesamiento* (unigramas), ratificando mediante este análisis la importancia de la fase de preprocesamiento.

[1]Son herramientas que miden la similitud entre dos árboles sintácticos en términos de sus subestructuras[43], donde pueden utilizarse para el análisis o la clasificación de frases con aprendizaje automático.

La desventaja básica de este proceso es la cantidad de trabajo que implica para poder llevar a cabo este análisis, pues la etapa de preprocesamiento involucre diversas tareas, sin embargo, cabe destacar que es viable la automatización de algunas tareas.

Es claro que el alcance de este proyecto permite llegar a analizar no sólo tweets, sino cualquier fuente de información que sea textual, ya que en este trabajo se han manipulado los datos buscando que su análisis sea factible.

4.4 Conclusiones del capítulo

Los trabajos que se han presentado en esta sección han permitido ratificar las nuevas demandas de almacenamiento y procesamiento de los datos que son generados a través de las actividades que realizamos diariamente, así mismo se identificó la importancia de realizar un análisis de estos grandes volúmenes de datos, para poder cumplir con este objetivo es necesario utilizar las herramientas adecuadas para estas nuevas demandas.

Adicionalmente al reconocimiento de la importancia del análisis de los datos, a través de este capítulo se ha logrado identificar los pasos que conforman la metodología de desarrollo que fue seguida para cada caso particular, comparando las tareas involucradas en cada una de ellas de tal modo de que se ha podido realizar una identificación de las similitudes entre los trabajos expuestos en esta sección.

Derivado de éste breve estudio se ha logrado identificar los pasos que conformarán la metodología que se seguirá para el desarrollo del proceso analítico que se implementará, los detalles de dichos pasos se abordarán en el capítulo 5.1

I have a cell phone that doesn't behave like a phone: It behaves like a computer that makes calls. Computers are becoming an integral part of daily life. And if people don't start designing them to be more user-friendly, then an even larger part of the population is going to be left out of even more stuff.

Alan Cooper

CAPÍTULO

5

Arquitectura solución y metodología utilizada para análisis Big Data

5.1 Elección de la metodología de desarrollo para la solución propuesta

A partir del capítulo 4 se han podido identificar las etapas de una metodología de desarrollo, las cuales son de vital importancia para poder llevar a cabo la concepción de este tipo de aplicaciones. Para la determinación de dichas etapas se han analizado las ventajas y el impacto que cada una de estas etapas brinda a etapas subsecuentes con base en cada uno de los trabajos que han sido presentados en dicho capítulo.

5.1.1 Definición de metodología de trabajo

Gracias a esta revisión se han seleccionado las fases que conforman la metodología que se utilizará para esta propuesta, las cuales se enlistan y detallan a continuación.

1. **FASE 1: Definición del propósito y alcance.**
   Esta fase es de gran importancia, tal como hace Raghupathi[41], a través de esta etapa se logra comprender el contexto de la situación que se pretende atender, de tal forma que se deja en claro qué es lo que se persigue con el proceso analítico.
   Esta etapa del proceso es fundamental debido a que gracias a lo que se defina en el alcance y propósito se podrá tener en claro cuales son los resultados finales esperados, esto a su vez involucra la identificación de las tareas que se deberán de llevar a cabo para cumplir con el objetivo. El objetivo general permitirá realizar la elección de las fuentes de información, así como de las herramientas que más se adecuen a la tarea que se desea llevar a cabo.
   Así mismo, esta etapa evitará la inversión de recursos en tareas que no tengan aportación con los objetivo planteados en la sección 1.4.

2. **FASE 2: Adquisición de los datos.**
   La adquisición de los datos es una etapa primordial, esta fase es implementada en [41], [9] y [42], a través de esta fase se obtiene el corpus que será analizado. A través de esta fase se establecerán las fuentes de información.
   En esta fase se debe de considerar que la información al provenir de Twitter no es estructurada, por lo cual se requerirá realizar la extracción y almacenamiento de información de manera adecuada acorde a las características.

3. **FASE 3: Preparación de los datos.**
   Tal como se indentificó con el trabajo de Agarwal et al., en [42], la preparación de los datos es una etapa fundamental ya que a través de esta se logra que el proceso analítico que se implemente (en este caso un análisis sentimental y la técnica de agrupaminento con lógica difusa) obtenga mejores resultados.
   En esta etapa se realizarán las manipulaciones requeridas con la finalidad de adecuar los datos para su análisis. A través de este proceso se podrá segurar la calidad de la información.
   Diversas tareas estarán involucradas, las cuales son:

- Filtración: de los datos mediante la cual será posible descartar los datos que no sean de interés para el análisis.

- Perfilado: analizar las características de los datos, decidir qué hacer con aquella parte de los datos que estén incompletos. La compresión de los datos suele ser una tarea recurrida en este tipo de ambientes ya que con la que se está trabajando tiene dimensiones grandes.

- Limpieza: homogeneización de formatos y representación de atributos, así como corrección de datos nulos y faltantes.

- Generación de metadatos: esto con el propósito de describir la información que se ha almacenado y guardar las fuentes de información, es decir, de donde fueron obtenidos los datos de tal manera que la identificación de los registros duplicados se viable.

- Transformación: los datos de origen se deben transformar en un formato que puede ser aceptado por el destino.

Se busca que al final de esta etapa, después de haber aplicado todas las actividades mencionadas en los puntos anteriores se obtenga como salida a los datos en forma adecuada, siendo entonces viable comenzar con el proceso analítico.

4. **FASE 4: Análisis de los datos.**
   A partir de las fases definidas previamente, se ha llegado a la etapa correspondiente al análisis de los datos, es decir, es hasta este punto cuando se comienzan a aplicar las técnicas de Minería de Datos, tales como son Regresión Lineal, análisis sentimental y una técnica de agrupamiento; para la obtención de un modelo que sea acorde con los objetivos que fueron planteados al inicio del proyecto de ambiente Big Data. El minado de los datos se puede realizar a través de diversas técnicas y métodos que en la actualidad se encuentran disponibles desde interfaces visuales, conjuntos de primitivas y lenguajes de consulta de minería de datos.[44]
   De acuerdo con Hrushikesha Mohanty[45] podemos identificar en este punto, básicamente cuatro tipos de análisis:

   a) BI (*Business Intelligence*), ROI (*Return On Investment*), reportes y cuadros para toma de decisiones.

b) Análisis descriptivo. Dividir a los clientes en segmentos y perfilar a los clientes que entran en esta categoría.

c) Análisis predictivo. Rentabilidad futura de un cliente, modelos de *scoring*[1] y pronosticar resultados como el *churn*[2] dentro de esta categoría.

d) Optimización. Los problemas de optimización y los escenarios hipotéticos se encuentran en esta categoría.

Para este trabajo se abordarán dos de los cuatro tipos mencionados anteriormente, el primero de ellos será un análisis de tipo descriptivo para el cual tendrá lugar un análisis sentimental. A través de este proceso analítico se podrá tener un panorama de la contexto actual de las opiniones de clientes en la red social.

Por otro lado, el segundo tipo de proceso analítico a realizar se encuentra comprendido en la categoría de análisis predictivo, a través de una regresión lineal se busca predecir el comportamiento de la popularidad de *Amazon Go*. Finalmente se implementará un algoritmo de agrupamiento permitirá conocer el comportamiento de las opiniones en Twitter.

5. **FASE 5: Evaluación de resultados.**

   Esta fase del proceso permite identificar si los resultados que se han obtenido en la fase inmediata anterior son de utilidad con respecto a los objetivos que se hayan definido en la primera etapa del proceso, en caso de que los resultados o modelos obtenidos cumplan con parámetros de calidad que serán establecidos de manera particular para cada método, entonces se considerará que los modelos obtenidos son aceptables.

   En caso de ser necesario, es viable regresar a fases anteriores convirtiendo esto en un proceso iterativo hasta que los resultados obtenidos cumplan con los objetivos que se han definido.

6. **FASE 6: Visualización de los resultados.**

   Finalmente, después de obtener conocimiento como resultado de la fase anterior, se deben de presentar los resultados de una manera inteligible[9], es decir, los resultados deben de ser entendibles para las personas que se encargarán de la toma de decisiones, ya que haber realizado todas las tareas involucradas en las fases previas carecerá de utilidad si los resultados

[1] Es el análisis de una información sobre un solicitante de un préstamo, en base al cual se recomienda luego su aprobación o rechazo.

[2] El *churn rate* o tasa de cancelación es el porcentaje de clientes o suscriptores que se dan de baja en un servicio.

no son entendibles, por ejemplo, si se ha realizado un análisis de ventas en un supermercado y los resultados del análisis son presentados de tal forma que solo pueden ser interpretados por la persona que se ha encargado de realizar dicho análisis, es claro que serán de poca utilidad para el proceso de negocio.

5.2 Arquitectura de la solución propuesta

Partiendo de la problemática que se ha presentado en el capítulo 1, y con base en las características mencionadas en el marco teórico (capítulo 3), ha sido posible la identificación de la necesidad de una solución analítica para un ambiente Big Data.

En esta sección se describe de manera detallada la arquitectura a través de las cual se desarrollará la aplicación que permitirá llevar a cabo el proceso analítico para cumplir con los objetivos presentados en la sección 1.4. Se presentan las etapas que se han identificado en la sección 5.1, explicando a mayor detalle cada una de estas etapas, así como los elementos que se verán involucrados para el funcionamiento de la aplicación analítica de tipo Big Data.

Es importante destacar el hecho de que en este capítulo no se presenta la implementación de los componentes que se describen, ya que la implementación se abordará en el capítulo 6.

A partir de la elección de la metodología de desarrollo presentada en la sección 5.1, para llevar a cabo la implementación de la solución que se propone en este trabajo se requerirán diversos módulos, estos se presentan de manera gráfica en la figura 5.1, identificando ahí los componentes involucrados para cada una de las fases.

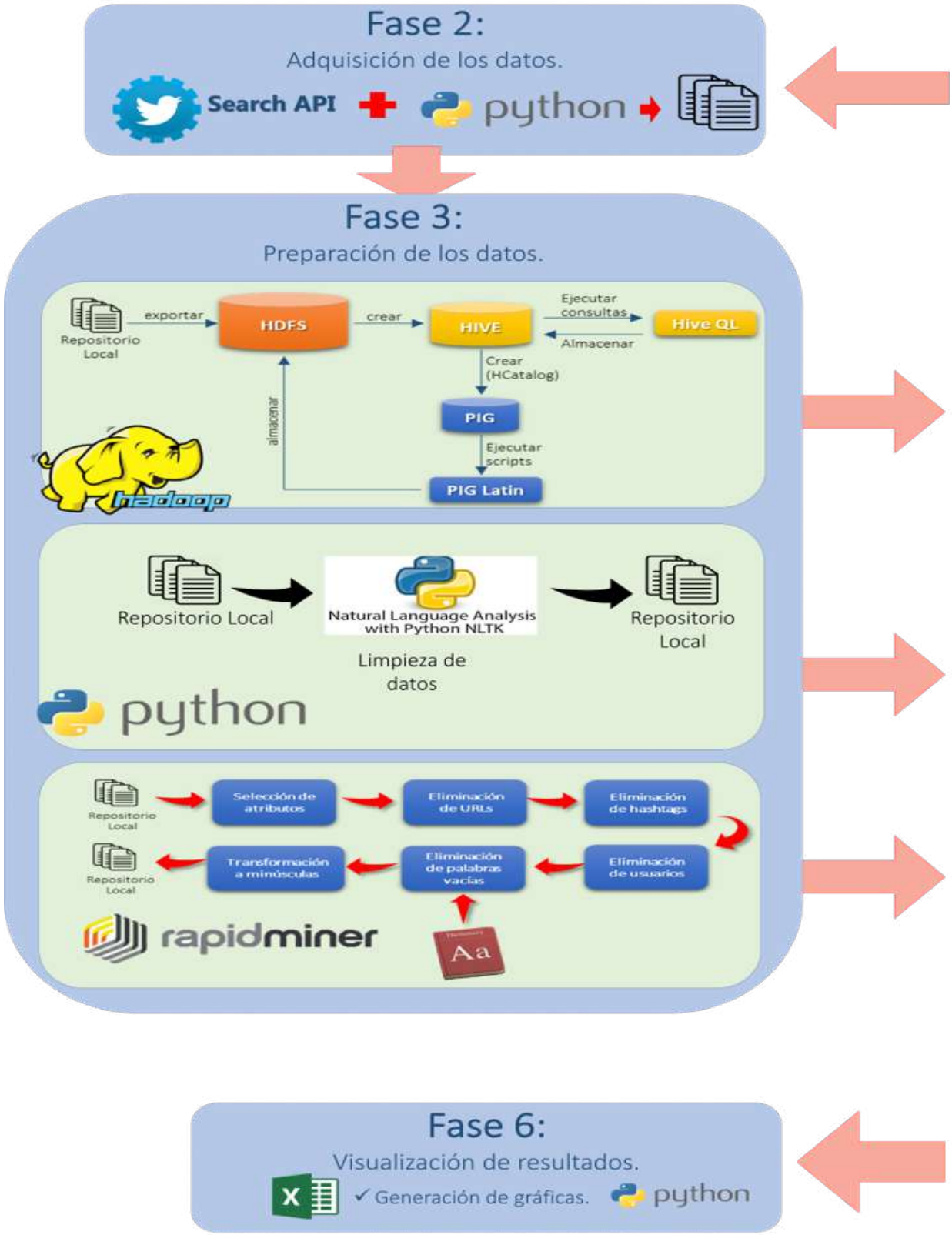


Figura 5.1: Arquitectura para la solución propuesta.
(Continúa en la págnina siguiente)

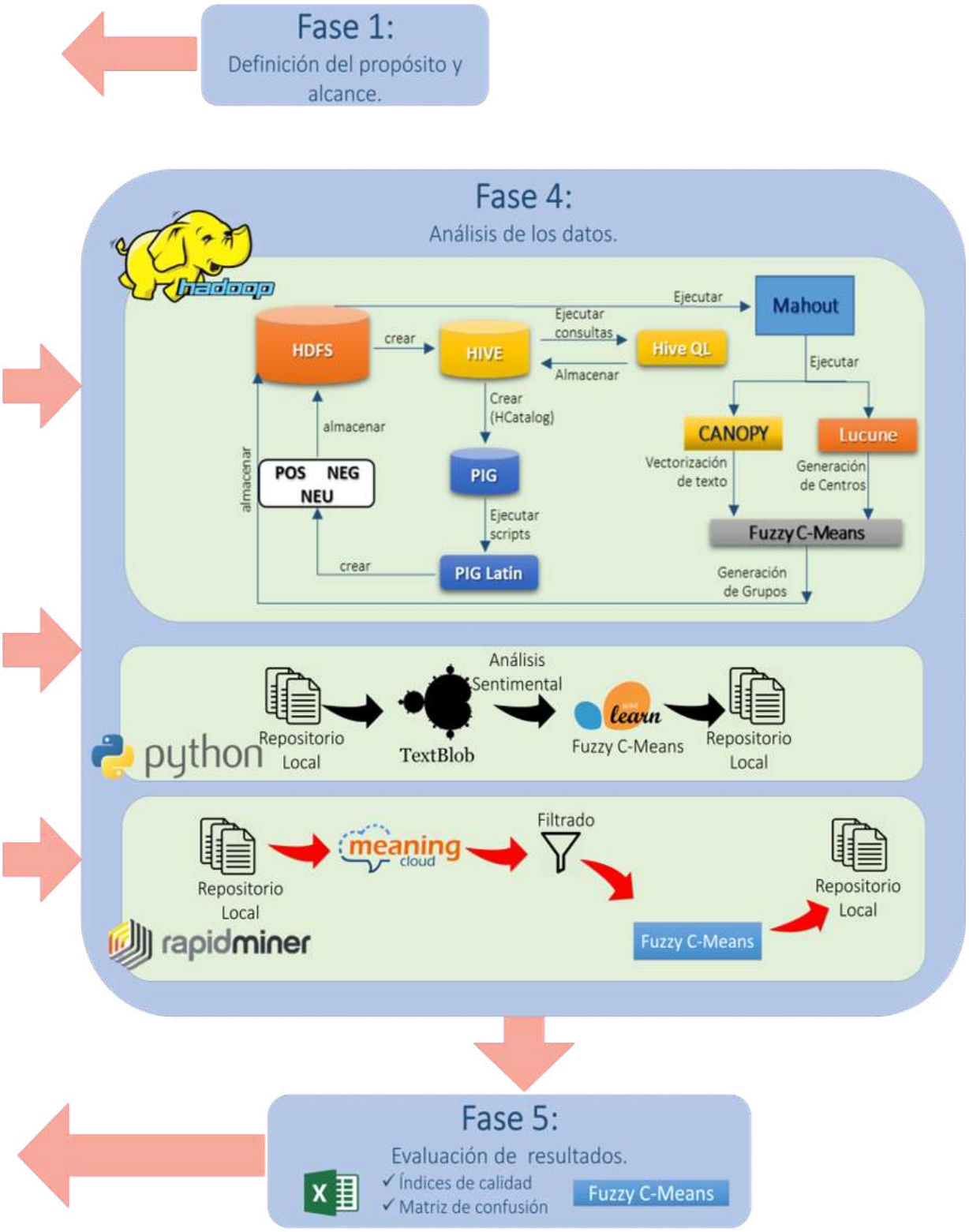


Figura 5.1: Arquitectura para la solución propuesta (continuación).

5.2.1 **Definición del alcance y propósito.**

Para esta primera fase básicamente se retoman los objetivos y limitaciones expuestos en el capítulo 1, de tal forma que se ha establecido de manera clara qué es lo que se persigue con la solución que se propone, y aunado a esto delimitar los alcances de dicha solución. Esta etapa se ha completado en el capítulo 1, en donde se habló sobre la problemática a abordar, especificando los objetivos que deberá de cubrir la aplicación analítica Big Data que se desea desarrollar.

5.2.2 **Arquitectura de la adquisición de datos.**

Para la siguiente fase *Adquisición de los datos*, a partir de la problemática identificada en el capítulo 1 ha sido posible llevar a cabo la selección de la fuente de información, la cual es acorde con la delimitación en la fase 1, es con la red social Twitter. Para lograr obtener las micro publicaciones realizadas por los usuarios de esta red social se utilizará la API de Twitter[1], la cual es una interfaz con un conjunto de subrutinas, funciones y procedimientos que ofrece servicios para poder ser utilizados como una capa de abstracción.

Los servicios que ofrece la API serán consultados a través de un script escrito en el lenguaje de programación Python, en dicho script se especificarán los parámetros necesarios que permitirán obtener el corpus de interés. Como salida de la ejecución de dicho script se obtienen los datos almacenados en un archivo, los cuales serán exportados a un HDFS y a RapidMiner[2], siendo estos ambientes de trabajo adecuados de acuerdo a la naturaleza de la problemática de interés. Todos los componentes involucrados en esta fase se ilustran en la figura 5.1.

5.2.3 **Arquitectura de la preparación de los datos.**

Después de obtener el corpus almacenado de tres formas diferentes, la primera será en el HDFS, la segunda en un directorio de trabajo ubicado el sistema operativo, y por último en RapidMiner, para la siguiente fase correspondiente a la preparación de los datos se encuentran involucrados diversos componentes, en la figura 5.1 se aprecian dichos componentes. El primero de ellos consiste en

[1]Disponible en *https://developer.twitter.com/*

[2]RapidMiner es un programa informático para el análisis y minería de datos. *https://rapidminer.com/*

la selección de atributos, esta tarea es importante porque a través de esta se logra filtrar aquellos atributos que no son de utilidad para las tareas analíticas, por lo cual lo más viable es descartarlos.

De acuerdo con lo mencionado anteriormente, para la implementación en el HDFS a través de HIVE[1] se generará un esquema para los datos, y realizando las proyecciones necesarias para el filtrado de los atributos, se almacenanarán los datos filtrados de nuevo en el HDFS. Mientras que para el caso de Rapid Miner, la herramienta cuenta con un módulo de filtrado (*select attributes*), el cual requiere que de manera previa se les haya establecido un rol a los atributos, por lo cual será necesario también utilizar el módulo correspondiente (*set role*). Finalmente, para la implementación en el sistema de archivos del sistema operativo se recurrirá a lenguaje Python, el cual cuenta con la capacidad suficiente para realizar el filtrado de los atributos de la manera deseada.

El formato que presenta el corpus obtenido en el módulo anterior y la descripción de los componentes de dicho formato se presentan a continuación:

username; date; retweets; favorites; text; geo; mentions; hashtags; id; permalink

Los campos de cada tweet que ha sido obtenido se encuentran separados por *punto y coma*. La definición de cada uno de estos atributos se detallan en la tabla 5.1.

Atributo	***Tipo de dato***	***Descripción***
username	string	Corresponde al nombre del usuario que ha realizado la publicación del tweet, no se debe de confundir con el ID del usuario, este campo presenta en claro el nombre del usuario, tal cómo puede ser localizado en la red social.
date	date	Correspondiente a la fecha en la que se realizó la publicación del tweet, este campo utiliza el formato *YYYY-MM-DD HH:MM*.
retweets	int	Despliega el número de veces que la publicación original ha sido compartida, por lo cual el valor mínimo de la cardinalidad de este atributo es cero y el límite superior no existe.

Continúa en la página siguiente.

[1] Capa de Hadoop para proporcionar agrupación, consulta, y análisis de datos.

Atributo	*Tipo de dato*	*Descripción*
favorites	int	Despliega el número de veces que la publicación ha sido marcada como "Favorita", por lo cual el valor mínimo de la cardinalidad de este atributo es cero y el límite superior no existe.
text	string	Contiene el texto que ha sido publicado por el usuario, está codificado en UTF-8, y en este se encuentran URLs, hashtags, emoticonos y usuarios (palabras que comienzan con el símbolo @).
geo	string	Contiene la ubicación geográfica del lugar en dónde se realizó la publicación del tweet; se encuentra expresada en términos de latitud y longitud, ambas en grados decimales (e.g. 19.397926, -99.018429).
mentions	string	Por cada cita que ha tenido la publicación original este campo contendrá los usuarios que han realizado una cita a dicha publicación, por lo cual habrá instancias en las que este campo esté vacío y otras con una gran lista de usuarios.
hashtags	string	El campo contiene el listado de todos los hashtags que fueron publicados en el tweet.
id	string	Muestra un ID de tipo numérico que la plataforma asigna a cada publicación de los usuarios, por lo cual este valor es único para cada tweet. Tal como es de esperarse, este campo contiene valores muy grandes (e.g. 956981110965141000).
permalink	string	Para cada publicación se obtiene una URL a través de la cual se puede consultar dicha publicación. Este campo está formado por el nombre del usuario y el ID de la publicación (e.g. *https://twitter.com/USER_NAME/status/956981110965141507*).

Tabla 5.1: Atributos de los tweets obtenidos a través de la fase de adquisición de datos.

Después de que se han concluido las tareas referentes a la filtración de atributos, el siguiente componente de esta misma fase consiste en la eliminación de una serie de palabras que tampoco serán de utilidad para los propósitos de esta solución, este componente se encargará de eliminar usuarios, hashtags y URLs, ya que estas cadenas de texto no serán de utilidad para el proceso analítico.

Para el caso de RapidMiner, esta tarea se realizará a través de módulos de tipo *replace* contenidos en dicho software, en estos módulos es posible indicar mediante el uso de expresiones regulares el tipo de palabras que deberá de eliminar del corpus.

En lo que respecta al ambiente Hadoop, mediante los frameworks HIVE y PIG será posible llevar a cabo esta tarea. Estas herramientas al igual que en el caso de RapidMiner emplearán expresiones regulares. Finalmente, para el ambiente en el sistema de archivos del sistema operativo se empleará Python, el cual cuenta con la biblioteca *regexp*[1], la cual proporciona funcionalidades tales que mediante el uso de expresiones regulares es viable la eliminación de cadenas de texto dentro del corpus.

El último componente de este mismo módulo corresponde a la transformación a minúsculas y eliminación de palabras vacías del corpus, tal como se ilustra en la misma figura 5.1, para realizar esta tarea, en el caso de RapidMiner se emplean dos módulos adicionales, *transform cases* y *filter stopwords english*, los cuales también se encuentran contenidos en dicha aplicación. El primer módulo se encarga de convertir todo el corpus en letras minúsculas, mientras que el segundo se encargará de eliminar las *palabras vacías*[2] que encuentre en el corpus. Por otro lado, para el ambiente Hadoop nuevamente se recurre al framework HIVE ya que este cuenta con las funcionalidades necesarias para llevar a cabo dicha tarea.

Para el último ambiente, correspondiente al sistema de archivos del sistema operativo, nuevamente se recurrirá al lenguaje Python, el cual a través de la biblioteca NTLK[3] y con el uso de un diccionario que se le proporcione, será viable la eliminación de las palabras vacías.

Es importante destacar que para la eliminación de las palabras vacías se debe de utilizar un diccio-

[1]Este módulo proporciona operaciones de correspondencia de expresiones regulares similares a las que se encuentran en Perl.

[2]Es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc.

[3]Es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural simbólico y estadísticos para el lenguaje de programación Python.

nario en el idioma con el que se va a trabajar, que en este caso será el idioma Inglés.

Finalmente, como se ha mencionado en el capítulo introductorio de este trabajo, la información puede provenir de diversas culturas, lo cual conlleva a que en el corpus se encuentren caracteres que no sean codificables. Dicha situación se presenta a pesar de que la obtención de los tweets se realice sobre aquellas publicaciones en el idioma Inglés, existirán publicaciones que contengan caracteres especiales, por lo cual se deberá de realizar un filtrado de aquellos caracteres que no son codificables en ASCII. Para realizar esta tarea en los ambientes de RapidMiner y Hadoop se empleará una expresión regular que realice dicho filtrado; mientras que para el ambiente en el sistema de archivo del sistema operativo se empleará Python debido a que este cuenta con una función que permite llevar a cabo dicha tarea.

5.2.4 **Arquitectura del análisis de los datos.**

Tomando como entrada el corpus, el primer componente a emplear corresponde a un análisis sentimental, este primer proceso analítico permitirá tener un bosquejo de la distribución de las opiniones de los clientes a través de la red social, siendo este un primer indicativo de interés para el grupo comercial.

Para llevar a cabo este análisis de opinión se requerirá trabajar únicamente con los campos *text* y *date* de los tweets que se han obtenido, y tras la aplicación de algoritmos de este tipo se obtendrá el corpus etiquetado con la opinión contenida en cada tweet.

Para la realización de esta tarea se tiene en consideración tres formas de implementar el proceso, por un lado para el caso de Hadoop se utilizará un algoritmo el cual a través de un diccionario ponderado se determinará la polaridad de cada publicación, mientras que para la herramienta RapidMiner se emplea un módulo comprendido dentro de dicha herramienta, cuya salida contiene una clasificación de cada tweet bajo una de tres posibles etiquetas: *negative*, *positive* o *neutral*. Para el último ambiente, se empleará Python que en conjunto con *textblob*[1] cuenta con la posibilidad de implementar este proceso analítico.

Este proceso analítico será de utilidad en dos aspectos, el primero de ellos permitirá conocer la

[1]Es un nuevo conjunto de herramientas de procesamiento de lenguaje natural de python, que se apoya en NLTK y Pattern, y proporciona módulos de minería de texto, análisis de texto y procesamiento de texto.

opinión de las publicaciones que se han obtenido y en segundo lugar se podrá trabajar en el siguiente componente de esta fase únicamente con las publicaciones que hayan sido etiquetadas como negativas.

De acuerdo con lo dicho anteriormente, la salida del análisis sentimental será una entrada directa para la última fase del diagrama presentado en la figura 5.1 (visualización de resultados), ya que a partir de estos resultados se podrán obtener gráficos y reportes que serán de utilidad para la toma de decisiones. Mientras que al mismo tiempo, esta misma salida del análisis sentimental también será la entrada para el siguiente componente de esta fase: filtrado de tweets negativos.

En lo correspondiente al filtrado de tweets negativos (ilustrado en la figura 5.1), se emplea un módulo *fliter examples*, el cual también es una de las herramientas dentro de RapidMiner; para el caso de Hadoop bastará con la ejecución de scripts en PIG[1]; mientras que para el ambiente de sistema de archivos del sistema operativo se ejecutará un script en python que realice dicho filtrado. Este filtro se encargará de descartar aquellos tweets que han sido clasificados como *positive* o *neutral* en el análisis sentimental, de tal modo de que los tweets clasificados como *negative* son almacenados, dando paso al siguiente módulo correspondiente al algoritmo de agrupamiento *K-Means difuso*, tal como se aprecia en la figura 5.1.

Este último componente de esta fase es otro proceso analítico, el cual consiste en un algoritmo de agrupación, que de acuerdo con Witten, Frank, Hall y Pál, *"es un método de clasificación en donde se buscan grupos de ejemplos que vayan de la mano o tengan características en común"*[46]. De tal modo que a través de este módulo será posible realizar la identificación de tendencias dentro del corpus y derivado del filtrado previo, estas tendencias serán negativas.

5.2.5 Arquitectura para la evaluación de resultados.

Después de haber concluido con la fase analítica, continuando con el flujo presentado en la figura 5.1, se procederá a realizar la evaluación de los resultados obtenidos por los dos componentes analíticos implicados en la fase anterior, de tal forma que se realizará una comparación entre los desempeños de los diversos algoritmos analíticos ejecutados (para cada ambiente utilizado, es decir, RapidMiner, Hadoop y el sistema de archivos del sistema operativo).

[1] Pig es una plataforma de alto nivel para crear programas MapReduce utilizados en Hadoop.

Para poder llevar a cabo esta fase, en lo que concierne al algoritmo de agrupamiento se empleará una métrica de validación externa, tal como se expuso en la sección 3.2.2.2. Por otro lado, la métrica de validación elegida para el análisis sentimental es la matriz de confusión, la cual es una herramienta que permite la visualización del desempeño de un algoritmo en cuestión.

Esta métrica realiza una comparación de valores que han sido clasificados correcta e incorrectamente para poder así establecer un criterio que permitirá decidir si el modelo obtenido tiene un buen desempeño, es decir, determinar si la tasa de error está dentro de valores aceptables.

5.2.6 **Arquitectura de visualización de resultados.**

A partir de la finalización de la fase anterior, se podrán generar reportes de los resultados de los algoritmos, de tal forma que se externará la preferencia por alguno en particular. Partiendo del hecho de que los modelos empleados cumplen con los criterios de aceptabilidad que se establezcan, entonces es viable pasar a la última fase presentada en la figura 5.1, la cual corresponde a la visualización de resultados, en la figura 5.1 se ilustran los componentes que permitirán completar esta fase.

Por un lado, los resultados que se han obtenido a través de RapidMiner pueden ser representados de manera gráfica a través de la misma herramienta mediante módulos adicionales que vienen comprendidos en el software, tal como se ilustra en la figura 5.1. Mientras que por otro lado, para el caso de Hadoop y el sistema de archivos del sistema operativo se contempla recurrir al lenguaje de programación Python, a través de la biblioteca *matplotlib*, la cual permite realizar la elaboración de gráficos de diversos tipos, tales como son: diagramas de pastel, gráficas de tendencias, gráficos de dispersión, gráficas de barras, entre otros.

5.3 **Conclusiones de metodología de desarrollo elegida y arquitectura a implementar.**

En este capítulo se ha identificado la metodología de desarrollo de aplicaciones Big Data, se definieron las fases que componen a dicha metodología, además, se ratificó la importancia y utilidad de cada una de las fases comprendidas dentro de esta metodología, se describieron las tareas que

involucran cada una de estas etapas y la manera que en conjunto permiten llevar a cabo un proceso analítico Big Data.

Por otro lado, a partir de esta metodología se han definido la arquitectura que se implementará como solución propuesta, detallando componentes que serán necesarios para poder llevar a cabo la implementación de la aplicación que permitirá realizar el análisis que atiende la problemática de interés.

Cada una de las fases identificadas emplea uno o más módulos, ya sean de uno o varios software, buscando que al trabajar en conjunto logren completar las tareas necesarias para alcanzar los objetivos de cada fase. Cabe hacer notar que el detalle de la implementación se presentará en el capítulo 6, mientras que el análisis de resultados de cada uno de los componentes a emplear serán presentados en el capítulo 7.

A calculator is a tool for humans to do math more quickly and accurately than they could ever do by hand; similarly, AI computers are tools for us to perform tasks too difficult or expensive for us to do on our own, such as analyzing large data sets or keeping up to date on medical research.

Oren Etzioni

CAPÍTULO

6

Implementación de la propuesta

En este capítulo se detallará la implementación de la arquitectura que ha sido introducida en la sección 5.2, presentando de manera puntual la implementación de cada uno de los componentes involucrados en la arquitectura ilustrada en la figura 5.1.

Partiendo desde la segunda fase correspondiente a la adquisición de datos, se explicará de forma precisa cómo se realiza la ejecución de los servicios ofrecidos por la API de Twitter. Posteriormente, se explicará la fase de preparación de datos, presentando la forma a través de la cual se realiza la selección de los atributos con los que se trabajará, así como la adecuación de los mismos para el proceso analítico correspondiente.

Posterior a que el corpus ha sido limpiado y preparado, en la etapa análisis de datos se presentarán los algoritmos sentimentales a utilizar (de acuerdo al ambiente en donde se ejecutaran), de tal forma que se tendrá el corpus etiquetado y partiendo de esto se detallará la manera en la que se realizará el filtrado para trabajar únicamente con aquellos tweets que han sido etiquetados como negativos; a partir de este nuevo corpus se podrá pasar a la técnica de agrupamiento con lógica difusa, especificando la manera en la que dicha técnica será implementada con cada una de las tres herramientas analíticas mencionadas en la sección 5.2.3.

Después de haber realizado las ejecuciones de los algoritmos mencionados anteriormente, se deberá proceder con la siguiente fase correspondiente a la evaluación de resultados, siendo aquí donde con mayor nivel de detalle se presentará forma en la que se realizará la evaluación del desempeño de los algoritmos utilizados, de tal forma de que se logre establecer un criterio de aceptación para los resultados obtenidos y al mismo tiempo definir un punto de comparación entre el desempeño de dichos algoritmos.

Con base en la elección de aquellos algoritmos con mejor desempeño se podrá pasar a la última fase de visualización de resultados, en la cual se explicará la forma a través de la cual se obtendrán los reportes y gráficos, los cuales permitirán responder a las cuestiones que han sido presentadas en la sección 1.6 de este trabajo, cumpliendo así con los objetivos para los cuales se propone esta solución analítica de ambiente Big Data.

6.1 Implementación de la adquisición de datos.

Tal como se presentó en la sección 5.2, para esta fase se utilizará la API de Twitter, la cual es la forma en que los programas informáticos se comunican para lograr el intercambio de información. Para lograr esta comunicación la API requiere que se establezca un *punto de conexión*, a través de este punto de conexión es como se logran establecer los protocolos para llevar a cabo el intercambio de información.

Para poder acceder a los servicios de la API de Twitter se tiene que registrar una aplicación, la cual al ser creada, de forma predeterminada solo puede leer información pública en Twitter. Para realizar el registro de una aplicación es necesario contar con una cuenta dentro de la red social, después de contar con dicha cuenta de usuario se deberá acudir a la dirección web `https://developer.twitter.com/en/apps`, en donde se desplegarán las aplicaciones que se encuentren actualmente registradas en la cuenta. Para este trabajo, se creará una nueva aplicación a través de la opción *create an app*, la cual redireccionará a un formulario en donde se deberán de llenar los campos correspondientes a *app name* (nombre de la aplicación), *application decription* (descripción de la aplicación), *Website URL* (URL del sitio web) y *Tell us how this app will be used* (Explicación de los usos que tendrá la aplicación). El nombre de la aplicación que se creará para este trabajo tendrá el nombre de *Análisis Big Data PCIC*.

Después de haber completado los campos mencionados anteriormente, el sitio redirigirá a los detalles de la aplicación que ha sido creada, estos detalles se presentan en la figura A.1 (revisar anexo A). Donde el sitio web en cuestión contiene la información clasificada con base en tres criterios:

- App details (Detalles de la aplicación).
- Keys and tokens (Llaves y tokens).
- Permissions (Permisos).

Para poder ejecutar los servicios de la API es necesario ingresar a la sección *Keys and tokens*, en donde se despliegan las llaves para realizar la ejecución de servicios de la API. Tal como se aprecia en la figura A.2, el sitio muestra dos cadenas de texto conformadas por caracteres alfanuméricos sensibles a mayúsculas, la primera cadena corresponde a una llave de acceso a la API (*API key*) y la segunda cadena corresponde a una llave secreta (*API secret key*), ambas llaves generadas son únicas para la aplicación y a través de este par de llaves se ejecuta un método de encriptación asimétrico entre la aplicación que ejecutará los servicios y la API.

Para terminar con la configuración de la aplicación es necesario generar el par de llaves de acceso (*access token* y *access token secret*). Para ello basta con seleccionar la opción *create* en la misma vista, dicha acción nos devolverá un par de cadenas de tipo alfanumérico, los cuales funcionan de manera similar al primer par de llaves obtenido.

En caso de ser necesario estos pares de llaves pueden volver a ser generadas, basta con seleccionar la opción *regenerate* presentada en la misma vista, esta opción eliminará las viejas llaves declinando todos los servicios que tengan configurados y generará un nuevo par para la ejecución de los servicios de la API.

Hasta este punto se cuenta con la configuración necesaria para acceder a los servicios de la API de Twitter. El siguiente paso consiste en implementar la ejecución de dichos servicios, y, una de las formas más sencillas de implementar es a través del lenguaje de programación Python.

Como primer paso se deberá crear un archivo de configuración, por ejemplo *config.py*, en el cual se especifiquen los dos pares de llaves que han sido creados previamente para la aplicación que accederá a los servicios de la API. El contenido de este archivo se presenta en el bloque de código A.1.

Una vez que se tiene el ambiente de configuración el siguiente paso será la implementación de las peticiones a la API. Tal como se mencionó en el primer capítulo de este trabajo, se identificó el hashtag *amazongo* como término idóneo para realizar la búsqueda de tweets, así que ese será la consulta que se enviará a la API.

El bloque de código A.2, el cual ha sido implementado en el lenguaje Python, lleva a cabo las peticiones a la API.

La salida de una ejecución del código descrito anteriormente se presenta en la figura A.3. En dicha ilustración se verifica que la ejecución del código terminó sin ningún problema (parte superior de la ilustración), así mismo se presenta el tiempo que tomó realizar la petición de la consulta hacia la API de Twitter. Cabe mencionar que con esta ejecución se recolectaron 5,522 tweets.

6.2 Implementación de la preparación de datos.

Después de que se ha logrado obtener el corpus, la siguiente fase consiste en la preparación de datos. Antes de continuar será necesario almacenar el corpus en la plataforma adecuada. Tal como se mencionó en la sección 5.2.3, se han considerado tres ambientes que serán adecuados para las fases posteriores, estos se enlistan a continuación:

- HDFS (Hadoop).
- Sistema de Archivos del Sistema Operativo.
- RapidMiner.

En las secciones siguientes se presenta para cada uno de estos entornos la forma en la que se preparan los datos. Los entornos mencionados son independientes, por lo cual, en cada uno de ellos se implementarán las tareas particulares necesarias para cumplir con los objetivos de esta fase. Esto permitirá en determinar el ambiente óptimo para desarrollar un análisis Big Data.

6.2.1 Preparación de datos con Hadoop.

Para este primer sistema será necesario contar con el ambiente de Hadoop instalado y configurado previamente. De tal forma, que se pueda realizar la carga del corpus directamente sobre el sistema

de archivos. Para realizar la instalación de Hadoop se puede visitar el sitio web `https://hadoop.apache.org/releases.html`, en donde se encontrarán las últimas distribuciones del software y deberá de elegirse la adecuada para la plataforma con la que se trabajará, para este proyecto se utilizará el sistema operativo Linux en su distribución CentOS 6.4, el cual se encuentra disponible en el sitio web `http://vault.centos.org/6.4/isos/x86_64/`.

Es importante mencionar el hecho de que Hadoop se ejecuta sobre Java, por lo cual será requisito previo contar con la configuración de dicho software, en caso de requerir realizar la instalación se puede visitar el sitio web `https://www.java.com/es/` para mayor información.

6.2.1.1 **Selección de atributos con Hadoop**

Para poder comenzar con la primera tarea de limpieza, será necesario realizar previamente la carga del corpus al sistema de archivo HDFS. Para lo cual, el corpus se encuentra en un directorio local y mediante el comando *hdfs dfs -put <path_archivo>* se lleva a cabo la carga al sistema de archivo. En la figura A.4 se muestra que el corpus ha sido cargado correctamente después de la ejecución del comando anterior.

Para proceder con la selección de atributos se cargarán los datos en HIVE[1]. Para ello, será necesario definir la tabla sobre la cual serán almacenados dichos datos. Este proceso es realizado mediante el bloque de código A.3. En la figura A.5 se muestra la ejecución del script nombrado *script1.hive*, el cual contiene dicho bloque código.

Después de que la ejecución del script A.3 ha concluido de manera exitosa, se verifica en HIVE que la carga de los datos y de todas las instancias del corpus se hayan cargado correctamente. En la figura A.6 se presenta la salida del conteo del número de registros en la tabla que se ha poblado.

Una vez que se tiene una relación sobre el corpus, implementar la selección de atributos será más sencillo. Basta con ejecutar una consulta sobre HiveQL que realice la proyección sobre el atributo *text*. La consulta que realiza esto es *select text from tweetsText;*.

En la figura A.7 se presenta parte de la salida de la ejecución de la consulta anterior, incluyendo el número de registros en la tabla.

[1]HIVE utiliza HiveQL, el cual es un lenguaje de consultas basado en SQL. Las consultas sobre HiveQL son traducidas automáticamente trabajos de tipo MapReduce, de tal modo que a través de este lenguaje es posible realizar manipulación de datos almacenados en el HDFS.

En este punto se ha completado correctamente la tarea correspondiente a la filtración de atributos, de tal modo que es viable continuar con la siguiente tarea concerniente a la eliminación de URLs, usuarios y hashtags.

6.2.1.2 **Eliminación de URLs, usuarios y hashtags en Hadoop.**

En esta segunda etapa se tiene como objetivo la eliminación de aquellas palabras que carecen de utilidad para el proceso analítico. Para llevar a cabo esta tarea en el ambiente de Hadoop se recurrirá nuevamente a uno de los frameworks disponibles dentro de la arquitectura de Hadoop, Apache HIVE. El cual es un herramienta que proporciona funcionalidades de agrupación, análisis de datos y consultas.

Para llevar a cabo la tarea de esta sección se utilizará un script en HIVE, donde mediante el operador *regexp_replace* se puede realizar el reemplazo de una cadena por otra. La identificación del patrón a cambiar es especificado a través de una expresión regular. La sintaxis de dicho operador es la siguiente:

regexp_replace('«expresión-regular»','«cadena-reemplazo»')

Donde *«expresión-regular»* consiste en la expresión regular que describirá a los patrones que permitan localizar las cadenas de texto que se desean eliminar. Por otro lado, *«cadena-reemplazo»* será la cadena por la cual se reemplazarán las coincidencias de los patrones identificados por la expresión regular.

En lo concerniente a la eliminación de URLs, resulta viable encontrar la expresión regular que permita identificar este tipo de cadenas sobre el corpus. De tal modo que la expresión regular a emplear en esta primera tarea será la siguiente `(http(s)?:(//)?)?(www.|WWW.)?[-+/[.][-a-zA-Z./$%a-zA-Z0-9]+#?_+()=%&!0-9]+.`

Para llevar a cabo la implementación del operador *regexp_replace* se utilizará una sentencia de tipo *update*. Cuando los datos fueron cargados a HIVE con el código A.3 se definió la tabla *tweetsText* como tipo *texto*. Este tipo de tablas solo permite realizar consultas de tipo *select*, por lo cual será necesario cargar los datos en un tabla que realice consultas de tipo *update*. Mediante el código A.4 se define y carga una tabla sobre HIVE que cuenta con las propiedades transaccionales requeridas.

Después de que el bloque de código A.4 se ha ejecutado de manera correcta se tiene el ambiente

listo para proceder a utilizar el operador *regexp_replace*. Para utilizar dicho operador, tal como se ha mencionado anteriormente, se empleará una consulta de tipo *update*, que actualizará el valor del campo *text* por la salida del operador *regexp_replace*.

En el bloque de código A.5 se emplea una consulta *update*, la cual invoca al operador *regexp_replace* para realizar la sustitución de las URLs por un valor nulo. Es decir, se elimina la cadena del texto. Esta nueva cadena se almacena en la misma tupla.

Después de haber completado la eliminación de URLs es prudente continuar con la siguiente tarea correspondiente a la eliminación de hashtags. Para lo cual, de manera similar como se realizó en el caso de las URLs, se empleará el operador *regexp_replace*. La expresión regular necesaria para identificar los hashtags sobre el corpus es `#()?[-a-zA-Z0-9_/*#$%&()!?=@]*`. En el bloque de código A.6 se presenta la consulta con la expresión regular correspondiente.

La siguiente tarea por realizar en esta fase consiste en la eliminación de los usuarios, los cuales se pueden identificar fácilmente, ya que estos comienzan con el símbolo arroba (@). Para llevar a cabo esta tarea se procederá análogamente a las dos tareas anteriores: se recurrirá a una expresión regular, y mediante el operador *regexp_replace* se realizará la eliminación de los usuarios. La expresión regular `@()?([-_a-zA-Z0-9])*` será utilizada para llevar a cabo esta tarea. En el bloque de código A.7 se encuentra la consulta necesaria.

Finalmente, derivado de la ejecución del flujo de trabajo anterior, lograron identificarse sobre el corpus caracteres que no son codificables en ASCII. Convirtiéndose esto en una nueva tarea que debe ser atendida en esta fase. Para llevar a cabo esta tarea nuevamente se empleará una expresión regular, para este caso dicha expresión regular eliminará aquellos caracteres que no son alfanuméricos, signos de puntuación ni caracteres especiales. Dicha expresión regular es la siguiente `[^a-zA-Z0-9""]+`.

Nuevamente a través de la ejecución de una sentencia *update* se utilizará el operador *regexp_replace* logrando de este modo la eliminación de aquellos caracteres que no son deseados. El código A.8 presenta la consulta que permite realizar esta tarea.

Después de que la consulta del código A.8 ha sido ejecutada, se procede a verificar que el flujo de trabajo que se ha realizado haya cumplido con la salida esperada. Por lo cual, en la figura A.8 se presenta parte del corpus, en donde se puede verificar que dicho flujo de trabajo anterior ha funcionado de manera correcta.

La verificación del funcionamiento correcto del flujo de trabajo descrito en esta sección permite proseguir con la siguiente tarea correspondiente a transformación y eliminación de palabras vacías.

6.2.1.3 **Transformación y eliminación de palabras vacías en Hadoop.**

Esta tarea involucra dos procesos a implementar. El primero de ellos, consiste en la transformación de los tweets en letras minúsculas; y la segundo de ellos en la eliminación de palabras vacías.

La transformación del corpus a letras minúsculas es una tarea de gran importancia, ya que en etapas posteriores facilitará la implementación del proceso analítico. Al momento de analizar un token (cadena de caracteres) únicamente se tendrá que hacerlo en letras minúsculas y no se tendrá que recurrir a todas las posibles combinaciones entre mayúsculas y minúsculas de los caracteres que conforman dicho token.

Para llevar a cabo la implementación de esta tarea se utilizará la herramienta PIG, cuyo lenguaje PigLatin contiene el operador *lower(string)*. Éste recibe como parámetro una cadena de caracteres, devolviendo como salida la cadena *string* que se le pasó como parámetro, pero ahora la cadena estará en letras minúsculas.

En el bloque de código A.9 se encuentran las instrucciones necesarias para realizar esta tarea.

Después de ejecutar el bloque de código A.9 en Hadoop se verifica que la salida del proceso es correcta, para lo cual en la figura A.9 se muestra parte del corpus.

El siguiente proceso a realizar consiste en la eliminación de las palabras vacías o *stop words* en Inglés, las cuales son palabras que carecen de significado como artículos, pronombres, preposiciones, etc. Para este ambiente en particular se requerirá un diccionario de este tipo de palabras, existen diversos repositorios en Internet, por ejemplo, `https://www.ranks.nl/stopwords` contiene colecciones de palabras vacías en más de cuarenta idiomas.

Después de que se ha seleccionado la lista de palabras vacías que se desean eliminar del corpus, se deberán de cargar en el HDFS para que puedan ser procesadas. Con el código A.10 se realiza la carga del archivo *stopwords_dictionary* a PIG, de tal forma que la relación *stopwords* contiene todas las palabras vacías seleccionadas.

Para llevar a cabo la eliminación de las palabras vacías sobre el corpus se empleará el bloque de

código A.11. En el cual mediante un operador de tipo *JOIN* se realiza la identificación de aquellas palabras del diccionario que se encuentran en el corpus, y estas son descartadas, por último se reconstruye y almacena el tweet resultante. Este proceso es explicado a mayor detalle en la sección A.2 del anexo A.

Cabe destacar, que el resultado de este proceso de eliminanción de palabras vacías es almacenado en el HDFS en el directorio *SALIDA_FASE_3*.

Posterior a la ejecución correcta del flujo ejecutado con Hadoop para esta tercera fase (preparación de los datos) de la figura 5.1, antes de proceder a la siguiente fase, correspondiente al análisis de datos, es necesario verificar que el corpus se encuentra adecuado, para lo cual en la figura A.10 se presenta parte del corpus después de que se ha sometido a todo el proceso descrito previamente.

Con base en la figura A.10 se puede establecer que el flujo de trabajo que se ha desarrollado para esta tercera fase (preparación de los datos) con el ambiente Hadoop se ha cumplido correctamente, y entonces el corpus está listo para proceder a la siguiente fase, la cual de acuerdo con el diagrama presentado en la figura 5.1 corresponde al análisis de los datos.

6.2.2 Preparación de datos a través de Python.

Para la implementación de esta tercera fase con Python, se requerirá contar con el lenguaje de programación Python en su versión 3.0 o superior, el cual es multiplataforma. El software puede ser instalado en ambientes Windows (32 y 64 bits), Mac OS, Linux y otros. Para su instalación basta con visitar el sitio web `https://www.python.org/downloads/`, en donde se encontrarán todas las distribuciones disponibles, y de acuerdo a la que se seleccione se presentarán las indicaciones para llevar a cabo la instalación correcta.

Después de tener el ambiente de trabajo listo solo se requiere tener el archivo con el corpus en el directorio en el que se desea trabajar.

6.2.2.1 Selección de atributos en Python

El primer paso para llevar a cabo la selección de atributos es la lectura del archivo de entrada. Este procedimiento es realizado mediante el bloque de código A.12. En este bloque de código se

realiza la lectura del archivo que contiene el corpus, así mismo, se define el archivo en el que será almacenada la salida.

En este mismo bloque, se realiza la lectura del corpus, separándolo por salto de línea ya que este es el delimitador de cada tweet. Posteriormente, mediante el caracter delimitador (punto y coma) se itera sobre cada tweet para seleccionar los atributos de interés. Este procedimiento es explicado a mayor detalle en la sección A.2 del anexo A.

En la figura A.11 se presenta la salida en consola de la ejecución del código A.12. En la misma imagen, en la parte inferior se presenta parte de la salida, de este modo se verifica que la salida cumple con lo esperado.

Ahora que se tiene el corpus filtrado, dejando únicamente a aquellos atributos de interés, es viable proseguir con la etapa eliminación de URLs, usuarios y hashtags de esta misma fase (preparación de datos).

6.2.2.2 **Eliminación de URLs, usuarios y hashtags en Python.**

Después de que se han filtrado los atributos, la eliminación de URLs, usuarios y hashtags será la siguiente tarea a abordar para la fase de preparación de datos. Esta tarea será realizada con el lenguaje de programación Python. Análogo a como se realizó en el ambiente de Hadoop, se utilizarán expresiones regulares.

En el bloque de código A.13, se presentan los comandos necesarios para realizar la eliminación de cadenas particulares, para llevar a cabo este procedimiento se utilizaron expresiones regulares, a través de las cuales se identificaron los tokens de interés.

De manera similar a como se implementó este proceso en el ambiente de Hadoop, se leyó el corpus que se obtuvo después de haber aplicado el filtrado de atributos, y se procedió a iterar sobre tweet. Mediante expresiones regulares (una para cada tipo de token que se deseaba eliminar) se fueron eliminando los tokens de interés sobre el corpus. Así mismo, en caso de que el corpus tuviera caracteres no codificables estos eran eliminados. Este proceso es explicado a mayor detalle en la sección A.2, en donde también se presentan las expresiones regulares empleadas para este proceso.

En la figura A.12 se aprecia la salida en consola de la ejecución del bloque de código A.13, y se muestra el conteo de las eliminaciones que se realizaron para cada una de las categorías (expre-

siones regulares empleadas).En la misma imagen pero en la parte inferior se presenta parte del corpus después de que ha sido sometido a este proceso de limpieza verificando que cumple con las características esperadas. Por lo cual, es viable recurrir a la última tarea correspondiente a la transformación y eliminación de palabras vacías.

6.2.2.3 **Transformación y eliminación de palabras vacías en Python.**

Para la transformación de letras mayúsculas por minúsculas y la eliminación de palabras vacías mediante el lenguaje de programación Python se utilizará el bloque de código A.14, en el cual se emplea la biblioteca NTLK tal como se mencionó en la sección 5.2.3.

En dicho bloque de código se realiza la lectura del corpus resultante de la eliminación de URLs, hashtags, usuarios y caracteres no codificables. Así mismo, se carga en el ambiente el diccionario de palabras vacías del NLTK, el cual está conformado por 179 palabras vacías.

Para cada tweet se realiza en primer lugar la transformación de este a letras minúsculas, para lo cual se emplea el operador *lower*, y la extracción de los tokens que conforman al tweet. Posteriormente, para cada tweet se ejecuta un *join* entre los tokens que conforman al tweet y los tokens del diccionario, descartando aquellos que coincidieron con el diccionario, y después se procedió a reconstruir el tweet con los tokens restantes. El proceso descrito anteriormente se detalla a mayor nivel en la sección A.2 del anexo A.

Finalmente, se verifica que la salida de la ejecución del código anterior no genere ningún error, para lo cual en la figura A.13 se presenta la salida en consola de la ejecución del bloque de código A.14. Tal como se aprecia, no existe ningún error durante la ejecución del proceso. Verificando el contenido en el archivo de salida en la misma imagen, en la parte inferior se presenta parte del corpus que se ha obtenido a través de este proceso, el cual cumple con las características esperadas para este proceso.

Después de haber completado todas las tareas correspondientes a la preparación de datos mediante Python, se considera que es viable proceder a la siguiente fase de la metodología presentada en la sección 5.1.1, pues esta fase ha sido completada satisfactoriamente gracias al empleo del lenguaje Pyhton y bibliotecas disponibles para este lenguaje.

6.2.3 **Preparación de datos a través de RapidMiner.**

Este ambiente analítico permite el desarrollo de procesos de análisis de datos mediante el encadenamiento de operadores a través de un entorno gráfico, siendo esta interfaz gráfica una gran ventaja sobre otras herramientas analíticas, gracias a ella RapidMiner se diferencia de Hadoop al desistir de los comandos a través de una terminal.

Para poder trabajar con este ambiente se debe de realizar la instalación previa del software, el cual cuenta con diferentes distribuciones, ya sea para Windows (versión de 32 y 64 bits), Mac (requiere Mac OS 10.8+) o Linux (requiere Java 8), tal como se ilustra en la figura 6.1. Para acceder a la distribución deseada es necesario ingresar al enlace `https://rapidminer.com/get-started/` en donde, en caso de no contar con una cuenta de usuario se tendrá que crear, ya que a través de esta cuenta de usuario se gestionará la descarga de la aplicación. Para este trabajo se contempló utilizar RapidMiner Studio 9.0.

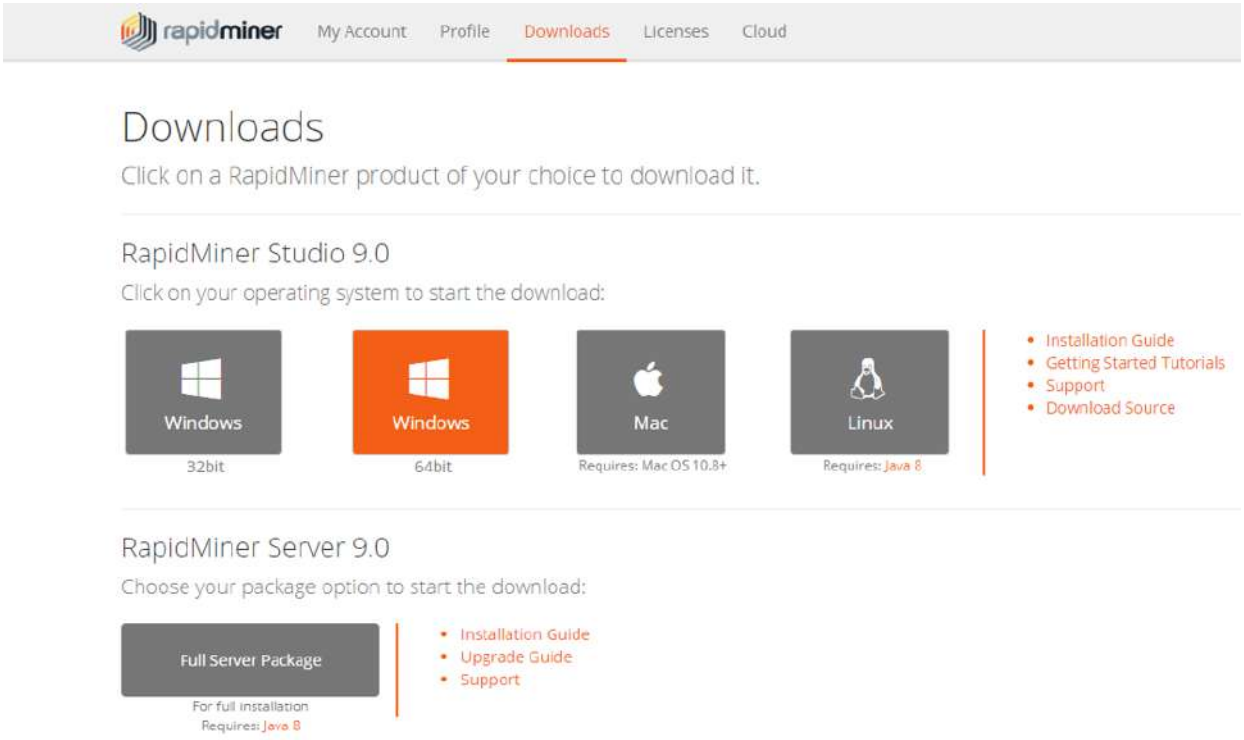


Figura 6.1: Distribuciones disponibles para RapidMiner Studio 9.0.

Después de contar con la configuración necesaria, el ambiente analítico cuenta con una sección correspondiente a repositorios, en donde se pueden almacenar flujos de trabajo, bases de datos y

salidas de flujos de trabajo. Es aquí donde se configurará un repositorio que contendrá el corpus que ha sido obtenido mediante el bloque de código A.2, el repositorio se llamará *corpus-original*.

6.2.3.1 **Selección de atributos en RapidMiner**

Una vez que se tiene el corpus en RapidMiner, lo siguiente es emplear los módulos comprendidos dentro de la herramienta analítica, creando la configuración necesaria que permita llevar a cabo la selección de atributos.

El primer módulo a emplear se llama *set role*, el cual permite establecer la forma en la que un atributo será manejado por otros módulos. La salida de este módulo será enviada al siguiente módulo *select attributes*, y este será el que permita realizar el filtrado para obtener únicamente los atributos de interés, que en este caso serán los campos correspondientes al texto y la fecha de publicación (*date* y *text*).

Finalmente, la salida de este flujo se almacena en un nuevo repositorio, cuyo nombre asignado es *tweets_filtrados* y, para realizarlo se emplea el módulo nombrado *store*. En la figura 6.2 se presenta el flujo descrito anteriormente.

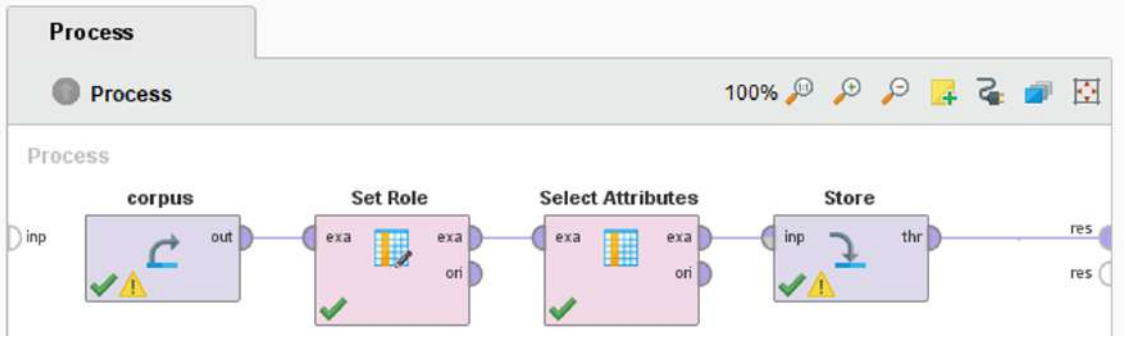


Figura 6.2: Diagrama de los componentes involucrados para filtrado de atributos con RapidMiner.

Para verificar que el flujo de trabajo anterior cumple con lo que se requiere se ha recurrido a la salida del mismo visualizador de resultados de RapidMiner, el cual presenta la salida del último bloque, y, que en esta caso consiste en el módulo *store*. Tal como se aprecia en la figura A.14, se muestra parte de la salida del proceso, ya que el corpus cuenta con 5,522 tuplas, y en la figura presentada solamente se muestran algunos de los registros que conforman dicho corpus. Sin embargo, a través de esto se puede verificar que se ha completado la tarea de filtración correctamente.

6.2.3.2 **Eliminación de URLs, usuarios y hashtags en RapidMiner**

Para esta siguiente etapa, correspondiente a la eliminación de ciertas cadenas de texto dentro del corpus, se requerirá emplear el módulo nombrado *replace*, el cual reemplaza cadenas contenidas dentro del atributo que se le indique por otro valor especifico, el módulo cuenta con dos posibilidades de implementación, la primera de ellas es a través de la búsqueda de una cadena en particular; y, la segunda opción es mediante el uso de una expresión regular.

Debido a que son cuatro diferentes tipos de expresiones que se desean eliminar dentro del corpus (usuarios, hashtags, URLs y caracteres no codificables en ASCII), se emplearán cuatro módulos de este tipo, esto con la finalidad de evitar crear expresiones regulares complejas, las cuales son más susceptibles a presentar errores en la validación de patrones.

En lo que respecta a la eliminación de URLs, a través de la expresión regular `(http(s)?:(//)?)?(www.|WWW.)?[-+/%a-zA-Z0-9]+[.][-a-zA-Z./$#?_+()=%&!0-9]+` se encontrarán las URLs dentro de cada tweet. Al módulo se le ha indicado que para cada coincidencia la reemplace por un valor nulo, es decir, eliminar la subcadena encontrada dentro de la cadena original.

Después de que se han eliminado las URLs del corpus, este flujo pasa al siguiente módulo, el cual es del mismo tipo (*replace*), pero ahora se empleará la expresión regular `#()?[-a-zA-Z0-9_/*#$%&()!?=@]*`, a través de la cual es posible realizar la identificación de los hashtags. De modo tal que, al terminar el parseo del corpus por este módulo ya no se tendrá ningún tipo de hashtag dentro del corpus.

La eliminación de las cadenas que corresponden a los usuarios de la red social (que comienzan a través del caracter @ seguido por el nombre de usuario) es la próxima tarea a realizar, para la cual se emplea nuevamente un módulo de tipo *replace*, pero ahora la expresión regular a utilizar será la siguiente `@()?([-_a-zA-Z0-9])*`.

Por último, tal como se mencionó previamente, aunque la recuperación de los tweets a través de la API de Twitter ha sido en idioma Inglés, el corpus contiene caracteres no codificables en ASCII. Por lo cual, se empleará otro módulo *replace* y mediante el uso de la expresión regular $[^a-zA-Z0-9]$ se eliminarán aquellos caracteres que no pertenecen al idioma Inglés.

Finalmente, el corpus que se obtenga después de aplicar todo el flujo de trabajo descrito previamente será almacenado en un nuevo repositorio de nombre *tweets_limpios*. Este proceso se ilustra

en la figura 6.3 en donde se presentan todos los módulos descritos previamente.

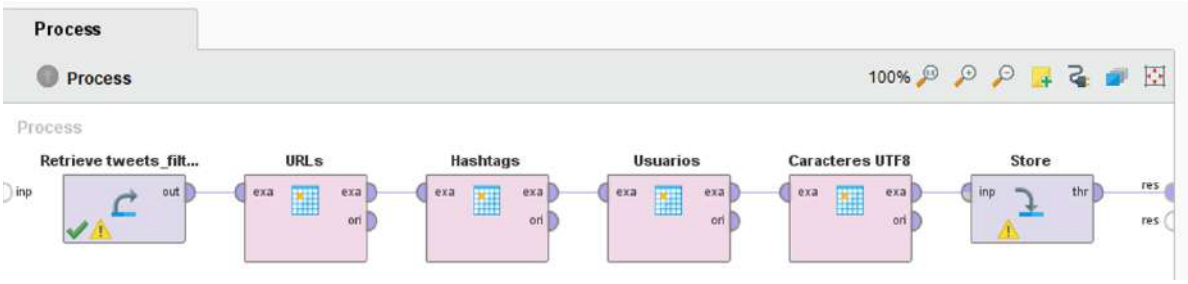


Figura 6.3: Diagrama de los componentes involucrados para la eliminación de URLs, hashtags, usuarios y caracteres no codificables en ASCII con RapidMiner.

Es importante verificar que el flujo empleado anteriormente haya funcionado de manera correcta. Para lo cual, en la figura A.15 se presenta parte de la salida del flujo anterior donde se aprecia que el corpus, ya no contiene ninguna de las cadenas que se han mencionado, ni tampoco caracteres que no son utilizados en el idioma Inglés.

6.2.3.3 **Transformación y eliminación de palabras vacías en RapidMiner.**

Para esta última tarea concerniente a la preparación de datos, se utilizará el corpus que fue obtenido en el flujo de trabajo anterior, y a este se le aplicará la transformación de las letras en mayúsculas que pudieran existir en el corpus por sus respectivas letras minúsculas. Para lograr esto, se empleará un módulo comprendido dentro de RapidMiner llamado *transform cases*, el cual no requiere ningún parámetro adicional para su configuración, basta con especificar el tipo de transformación que se desea realizar (a minúsculas en este caso), la fuente de los datos y redirigir la salida de la transformación.

Posteriormente, el flujo de trabajo continúa con un módulo nombrado *filter stop words*. En la herramienta existen diversos módulos de este tipo, diferenciándose únicamente por el idioma en el que trabajan. Sin embargo, si se requerirá de un módulo adicional previo a los dos módulos descritos anteriormente, ya que el módulo de eliminación de palabras vacías, y el de transformación de letras mayúsculas (*transform cases*) difieren al resto de los módulos que se han trabajo previamente, pues los módulos previos manejan como entrada y salida un flujo único o una muestra de datos, mientras que los módulos *transform cases* y *filter stop words* tienen la singularidad de que procesan datos de tipo *documento*, por lo cual es necesario recurrir al módulo nombrado *extract*

document ya que este módulo se encarga de convertir el flujo de trabajo en un formato de tipo *documento*, de tal forma que el flujo de trabajo queda tal como se ilustra en la figura 6.4.

Es importante resaltar el hecho de que se ha empleado el módulo *Filter stop words* correspondiente al idioma Inglés, el cual se aprecia en la última parte del flujo de trabajo (extremo derecho) presentado en la figura 6.4.

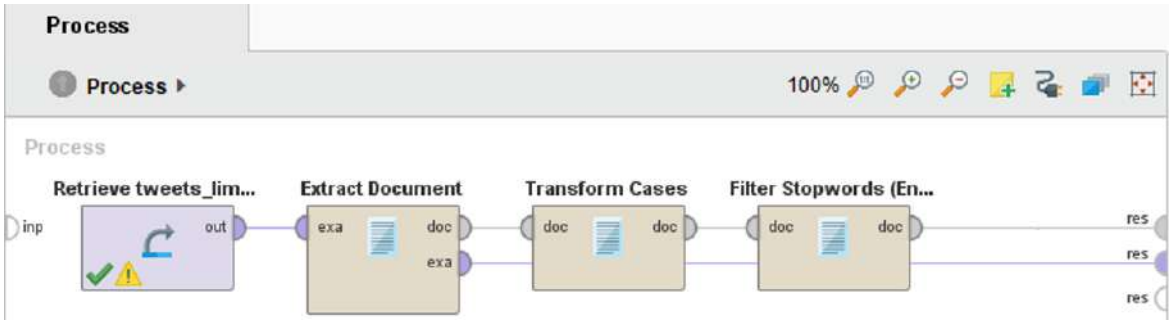


Figura 6.4: Diagrama de los componentes involucrados para *transformación y eliminación de palabras vacías* con RapidMiner.

De manera análoga en los dos flujos de trabajo previos de esta misma fase (selección de atributos y eliminación de URLS, hashtags, usuarios y caracteres no codificables), se debe verificar que la salida del flujo de trabajo presentado en la figura 6.4 sea correcta. Por lo anterior, en la figura A.16 se presenta una muestra en la que se compara el texto antes de entrar al flujo de trabajo de la figura 6.4 y la salida que es obtenida después de que dicho tweet ha sido procesado.

Tal como se observa en la figura 6.4, el texto a analizar ya se encuentra limpio de palabras vacías y en letras minúsculas. De este modo, se ha completado la tercera fase de la metodología correspondiente a la preparación de los datos.

6.3 Implementación del análisis de datos.

Teniendo listo el corpus el siguiente paso de la metodología presentada la sección 5.1.1 consiste en el análisis de los datos. Tal como se indicó en la sección 5.2, se realizarán dos tipos de análisis. El primero de ellos será de tipo descriptivo, el cual consistirá en un análisis de sentimientos (o de opinión); mientras que el segundo de ellos será un algoritmo de agrupamiento con lógica difusa.

6.3.1 Implementación de análisis sentimental

Como se ha mencionado previamente en la sección 5.2.4, para esta fase analítica se someterá el corpus a un análisis sentimental, el cual básicamente consiste en analizar cada tweet buscando determinar una etiqueta para el tweet a partir de la polaridad contextual de este. Al final de este procesamiento se tendrá el corpus etiquetado. Cada tweet tendrá una de tres posibles etiquetas. Las posibles etiquetas son: positivo, negativo o neutral.

6.3.1.1 Análisis sentimental implementado con Hadoop

La implementación del proceso analítico descriptivo con Hadoop consistirá en una evaluación sentimental para cada una de las palabras que conforman cada tweet. Para llevar a cabo dicho proceso, de acuerdo con la arquitectura presentada en la figura 5.1 (la parte izquierda de la fase 4 del diagrama corresponde al análisis sentimental), se utilizarán los framework HIVE y PIG.

Como primer paso será obtener un diccionario sentimental, este se ha obtenido desde el sitio web `https://goo.gl/A16HxE`, y está conformado por 2,477 palabras pertenecientes al idioma Inglés. Este diccionario tiene relaciones binarias de tipo «palabra, valor_sentimiento». Es importante mencionar que el dominio del campo «valor_sentimiento» se encuentra comprendido por los valores enteros entre -5 y 5, en donde -5 significa que la palabra tiene una connotación muy negativa y por el contrario, un valor de 5 implica que la palabra tiene una connotación muy positiva. En la figura A.17 se presenta parte del contenido del diccionario, en donde se corrobora lo mencionado anteriormente.

Posteriormente, se deberá de cargar el diccionario en HIVE y crearle su respectivo esquema. Esto implica la creación de una tabla en HIVE, y se lleva a cabo mediante el bloque de código A.15.

La figura A.18, mediante la ejecución de una consulta que devuelve los primeros cinco registros de la tabla *sentiment_analysis*, muestra parte del diccionario que se ha cargado. A través de esta figura se verifica que la carga ha sido correcta, respetado el formato que contenía el archivo original.

Después de tener listo el diccionario sentimental en HIVE, el siguiente paso es la ejecución del proceso analítico. Para llevar a cabo la implementación de dicho proceso se utilizará el framework PIG, a través de un script en PigLatin será viable realizar dicho proceso analítico.

Este proceso analítico se realizará mediante el bloque de código A.16, donde a partir del corpus adecuado (que previamente fue almacenado dentro del HDFS en el directorio SALIDA_FASE_3), se procede a realizar el análisis sentimental con ayuda de PIG.

Como primer paso, cada tweet contenido dentro del corpus se obtendrán los tweets que conforman cada tweet. A partir de estos tokens mediante la ejecución de un *join* se identifican los tokens que se encuentran dentro del diccionario, asociando a cada uno de estos tokens su respectivo valor (con base en el diccionario). Posteriormente, se realiza la regeneración de tweet, seguido de la suma de los valores correspondientes a cada token, de tal modo que se obtiene una evaluación del tweet, a partir de la cual se determinará la polaridad del tweet.

Estos tweets etiquetados son almacenados nuevamente dentro del HDFS, pero ahora el directorio SALIDA_FASE_4. Adicionalmente, en la última parte de este mismo bloque de código se realiza el filtrado de aquellos tweets que han sido etiquetados como ñegativos'y son almacenados en el directorio TWEETS_FILTRADOS. Este último paso será de utilidad para el siguiente proceso analítico, correspondiente al algoritmo de agrupamiento con lógica difusa.

El proceso descrito anteriormente se presenta a mayor detalle en la sección A.3 del anexo A.

Es importante verificar que la ejecución del script con el código A.16 no tuvo ningún problema y que la salida de este ha sido correcta. Por lo cual en la figura A.19 se presenta parte de la salida de la ejecución del comando *hdfs dfs -cat /SALIDA_FASE_4/part-r-00000*, en donde se aprecia que dicho archivo contiene la salida del análisis sentimental.

Por otro lado, en la figura A.20 se presenta parte del contenido del directorio *TWEETS_FILTRADOS*, el cual se ubica en el HDFS, tal como se esperaba. Se ha creado un directorio por cada tweet que ha sido etiquetado como negativo, y dentro de cada directorio se encuentra el texto correspondiente. El nombre que se ha asignado para la creación de cada directorio corresponde al identificador del tweet.

6.3.1.2 **Análisis sentimental implementado con Python**

El lenguaje de programación Python cuenta con la biblioteca *TextBlob*, la cual está basada en NLTK. Una de las principales diferencias entre NLTK y TextBlob es que la segunda ofrece funcionalidades adicionales a las de NLTK, tales como análisis de sentimientos, etiquetamiento de

palabras, extracción de frases sustantivas, etc.

TextBlob ofrece la ventaja de que las variables de este tipo son tratadas como cadenas de texto con Python, lo cual permite aprovechar de las funcionalidades que brinda el lenguaje, pues no requiere realizar conversiones entre tipos de datos.

Para llevar a cabo el análisis sentimental con Python se empleará dicha biblioteca. En el bloque de código A.17 se presentan las instrucciones que realizarán el análisis sentimental al corpus.

Lo que se realiza en este bloque de código invocar a la biblioteca TextBlob, la cual cuenta con su analizador sentimental. Se carga el corpus en el ambiente, y se itera para cada tweet, somentiéndolo al analizador sentimental, de tal forma que se obtiene una etiqueta para cada tweet. Estos son almacenados con base en la clasificación asignada. Este proceso se explica a mayor detalle en la sección A.3 del anexo A.

En la figura A.21 se presenta parte de la salida en consola de la ejecución del código A.17, se aprecia que dicha ejecución no presentó errores y además se han seleccionado algunos tweets de manera aleatoria con el propósito de verificar que la etiqueta que se ha asignado es correcta de acuerdo con el contenido del mismo.

También es importante realizar una revisión de que el proceso analítico llevado a cabo por el bloque de código A.17 ha funcionado correctamente, y que además ha sido almacenado acorde a lo planificado. Por lo cual, en la figura A.22 se presenta parte del corpus, donde se aprecia que éste ahora se encuentra etiquetado.

Finalmente, después de haber verificado que el trabajo realizado con Python ha cumplido con la sección analítica descriptiva, es viable continuar con el siguiente proceso analítico a implementar con Python, el cual corresponderá a la técnica de agrupamiento con lógica difusa.

6.3.1.3 **Análisis sentimental implementado con RapidMiner**

La implementación de este proceso analítico descriptivo en RapidMiner resulta ser poco complicado de implementar, similar a como se realizó en la sección 6.2.3, la herramienta cuenta con un módulo que permite realizar esta tarea. Este módulo se llama *analyze sentiment*, el cual ejecuta los servicios ofrecidos por la API de AYLEN (disponible en `https://aylien.com/text-api/`).

Este módulo obtiene una evaluación sentimental del texto, puede proporcionarnos información valiosa sobre las emociones y la perspectiva del autor: si el tono es positivo, neutro o negativo, y si el texto es subjetivo (es decir, si refleja la opinión del autor) u objetivo (es decir, si expresa un hecho).

Para utilizar este módulo basta con colocarlo en un nuevo proceso, así como la fuente de datos, la cual será el corpus que se obtuvo de la sección 6.2.3, y se encuentra adecuado para ser sometido a este proceso analítico. La configuración del módulo requiere únicamente la especificación del atributo a analizar, que en este caso, como el corpus que se ha utilizado ha sido adecuado previamente en la fase 3 (*preparación de los datos*) se utilizará el atributo correspondiente al texto.

En la figura 6.5 se ilustra la configuración necesaria descrita anteriormente. Adicionalmente, para que el resultado del proceso analítico descriptivo sea almacenado para etapas posteriores se agregó un módulo de tipo *store* el cual permitirá almacenar el corpus etiquetado en un repositorio local.

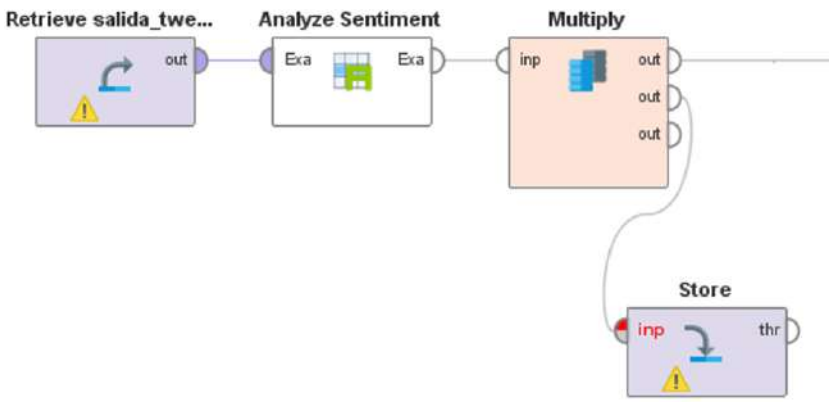


Figura 6.5: Diagrama de componentes involucrados para análisis sentimental con RapidMiner.

Después de ejecutar el flujo de trabajo descrito, como parte del resultado de este se aprecia en la interfaz gráfica los resultados que se han obtenido. En la figura A.23 se presenta parte de esta salida, garantizando así que el análisis sentimental se ejecutó de manera correcta.

Teniendo el corpus etiquetado se concluye el proceso analítico de esta sección implementado en el ambiente de RapidMiner. Ahora resulta viable proceder con la siguiente tarea analítica de agrupación, y posteriormente con la evaluación de resultados que se han obtenido.

6.3.2 Implementación de agrupamiento K-Means con lógica difusa

El siguiente proceso analítico a implementar consiste en el método de agrupamiento K-Means con lógica difusa, en la literatura comúnmente se encuentra como *Fuzzy K-Means*. A través de este algoritmo se buscarán patrones en los datos sin tener una etiqueta específica, es decir, se pretende la identificación de las problemáticas expresadas por los clientes.

Esta técnica, de manera similar a como se realizó con el análisis sentimental, será implementada en los tres entornos de trabajo que se han manejado previamente.

Antes de continuar con la implementación del algoritmo en cuestión se destaca el hecho de que a partir de este punto, se trabajará con el corpus que contiene únicamente a aquellos tweets que fueron etiquetados como negativos, para cada ambiente se empleará el corpus resultante de la implementación del análisis descriptivo realizado previamente en cada uno de estos entornos.

Para la ejecución de este método, el primer aspecto que se debe atender es la determinación del número de grupos con el que se realizará la ejecución del método de agrupamiento en cuestión, para lo cual se empleará el método presentado en la sección 3.2.3: *El codo*.

Mediante el bloque de código A.18 se implementará el proceso en cuestion. Después de cargar en el ambiente las bibliotecas necesarias, lo siguiente que se hace es definir una función que reciba como parámetros el texto a agrupar y el número de grupos que formará. Esta función devolverá los grupos formados, así como información relacionada a la ejecución.

Después de tener definida la función que ejecuta el algoritmo *fuzzy C-Means* lo que prosigue es iterar en un intervalo de 2 a 20 grupos, obteniendo para cada ejecución el valor de la inercia. A partir de este grupo de valores se obtiene la gráfica que permite determinar el número óptimo para ejecutar el algoritmo de agrupamiento en cuestión.

El resultado de la ejecución del bloque de código anterior se ilustra en la figura 6.6, donde se puede apreciar que el punto que se está buscando se encuentra localizado cuando el número de grupos es igual a 3, ya que partir de este valor se observa que el comportamiento de la variación de la inercia es asintótico. Por ello la atención se centra en intervalo que se presenta en dicha gráfica intervalo. Este proceso se explica a mayor nivel de detalle en la sección A.3 del anexo A.

Cabe mencionar que, de continuar analizando el comportamiento de la gráfica cuando el número

de grupos es mayor existe la posibilidad de localizar otro cambio drástico en la variación de la inercia. Sin embargo, utilizar este nuevo número mayor de grupos para la ejecución del algoritmo K-Means con lógica difusa llevará a encontrar subgrupos de los 3 grupos originales que han sido determinados a partir de la identificación del primer cambio drástico dentro de la curva.

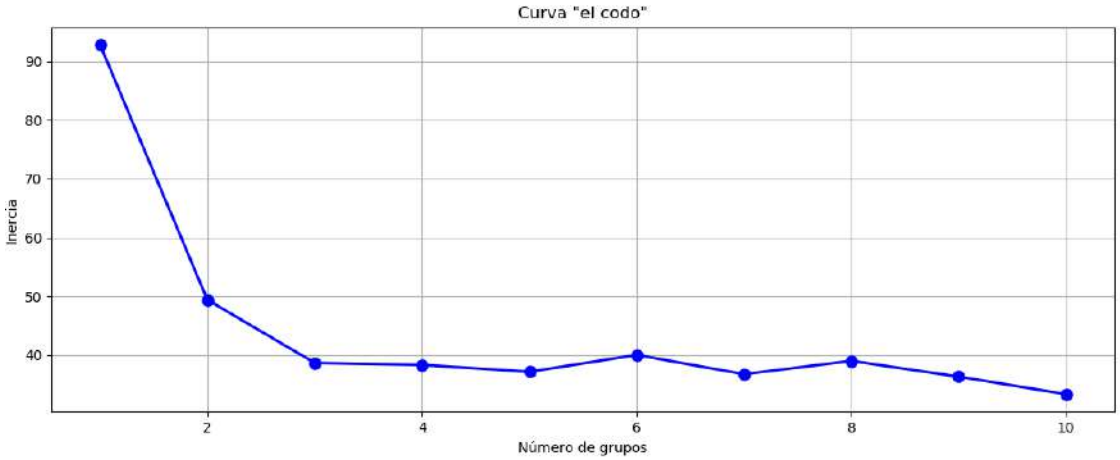


Figura 6.6: Graficación de la curva *el codo* para la técnica de agrupamiento K-Means con lógica difusa.

6.3.2.1 Implementación de algoritmo de agrupamiento K-Means con lógica difusa con Mahout

Para la implementación del algoritmo *Fuzzy K-Means* en Hadoop será necesario emplear el framework de la familia Apache *Mahout*, el cual brinda grandes funcionalidades analíticas, entre ellas se encuentra la implementación del método de agrupamiento de interés.

Como primer paso para la implementación se deberá configurar el ambiente necesario para poder utilizar dicha herramienta, por lo que adicionalmente dicha herramienta será descargada desde el sitio web `https://mahout.apache.org/general/downloads`, en donde se encuentran los enlaces para descargar los binarios necesarios, así como las adecuaciones requeridas para el ambiente. Este framework opera en conjunto con el HDFS y Java, ambas herramientas ya se encuentran configuradas como parte del proceso implementado en la sección 6.2.1.

Teniendo listo el ambiente, así como el corpus resultante del proceso descriptivo realizado en la sección 6.3.1.1, y después de que éste ha sido filtrado conservando únicamente a aquellos tweets etiquetados como negativos, es viable dar comienzo al siguiente proceso analítico.

La ejecución de Mahout se realizará directamente en la línea de comandos, siendo el primer paso la definición y carga en el ambiente la variable que indique el directorio en donde se localizan los ejecutables de Mahout. Esto se realiza a través del bloque de código A.19.

Debido a que el método de agrupamiento en cuestión emplea valores numéricos en su ejecución, deberá de obtenerse una representación numérica del corpus. Para lo cual se empleará la técnica TF-IDF, la cual se presentó en la sección 3.2.2.1, de tal forma que se logre generar un vector de características para cada tweet.

Para llevar esto a cabo con Mahout, se requiere crear una secuencia de archivos con el corpus. A través de esta tarea Mahout transforma el corpus en un formato perteneciente a una clase de Hadoop, mediante la cual permite generar pares de tipo *«llave, valor»*, siendo justamente este el formato que utiliza Mahout para la creación de los vectores en cuestión.

Para realizar esta creación de secuencias de archivos se utiliza el bloque de código 6.1, el cual ejecuta el binario de Mahout y con el comando *seqdirectory*, se realizan la transformación en cuestión. Como parámetros de ejecución del comando, se encuentra en primer lugar el directorio con el corpus (dentro del HDFS), el cual se ha indicado con la opción *-i*, así mismo se debe de definir el directorio en el HDFS en donde se almacenará la salida, esto se logra especificar mediante la opción *-o*. Por otro lado, el parámetro *-c* recibe la codificación en la que se encuentra el corpus, en este caso se trata de UTF-8. El parámetro *-chunk* define el tamaño de los bloques en los que la salida será almacenada, para esta ejecución se han elegido bloques de 5 Mb. Finalmente, el parámetro *-ow* permite sobrescribir los resultados.

```
1 mahout seqdirectory -i TWEETS_FILTRADOS -o Agrupamiento/tweets-seq -c UTF-8
    -chunk 5 -ow
```

Código 6.1: Creación de archivos de secuencia para Mahout.

En la figura A.24 se ilustra la ejecución del comando *seqdirectory*, en dicha figura se muestra como parte de la salida los parámetros con los que se está ejecutando el comando.

Después de ejecutar el comando *seqdirectory* se puede verificar que se encuentre la salida en el HDFS, para lo cual basta con ejecutar el comando que muestre el contenido de directorio *Agrupamiento*, tal como se aprecia en la figura A.25.

Teniendo los archivos de secuencia listos se procede con la creación de los vectores de características a través de los cuales se obtendrá una representación numérica del corpus. Para realizar dicha representación se empleará la técnica TF-IDF expuesta en la sección 3.2.2.1. Para llevar a cabo esta transformación Mahout cuenta con el comando *seq2sparse*.

En el bloque de código 6.2 se presenta el comando en cuestión, especificando los parámetros necesarios.

```
1 mahout seq2sparse -i  Agrupamiento/tweets-seq/ -o Agrupamiento/tweets-vectores
    -ow -chunk 100 -x 90 -seq
```

Código 6.2: Creación de vectores de características para Mahout.

El directorio de entrada es el primer parámetro del comando *seq2sparse* y se encuentra especificado por la opción *-i*, el cual contiene los archivos de secuencias creados previamente. El parámetro *-o* permite especificar el directorio en donde serán almacenados los vectores generados por este proceso.

Este comando permite sobrescribir lo resultados en caso de que existan previamente, esto se realiza a través de la activación del parámetro *-ow*. Por otro lado, la definición del tamaño de los bloques donde se almacenará la salida se establece con el parámetro *-chunk*, el tamaño se especifica en Mb. El siguiente parámetro *-x*, permite definir el porcentaje máximo que puede aparecer un término sobre todos los tweets, lo cual permite eliminar términos de alta de frecuencia. La conservación del formato como una secuencia es indicado mediante la opción *-seq*.

La ejecución del comando anterior se aprecia en la figura A.26, en cuya primera línea se encuentra el comando descrito anteriormente.

Después de que la ejecución del comando *seq2sparse* ha concluido de manera correcta, se procede a verificar el directorio que contiene los vectores que han sido creados. En la figura A.27 se presenta el contenido de dicho directorio, y además se puede apreciar que con los vectores se han creado otros directorios adicionales empleados durante la ejecución de este proceso.

Contando con los vectores de características listos se puede proceder con la ejecución del algoritmo de agrupamiento K-Means con lógica difusa. Sin embargo, es de suma importancia tener los centros (propuestos de manera inicial) en el formato requerido por el algoritmo. Para atender esto,

una de las técnicas que ofrece Mahout es la técnica *Canopy*, la cual es una técnica de agrupamiento. Este algoritmo suele denominarse como algoritmo de pre-agrupamiento, es muy rápido en su ejecución y tiene como principal ventaja que no se requiere especificar los centros iniciales para su ejecución debido a que la selección de dichos centros es realizada por el propio algoritmo, lo cual lo diferencia de otros algoritmos implementados en Mahout, tales como K-Means, Spectral K-Means o Fuzzy K-Means. Al terminar la ejecución de este algoritmo se crea un directorio en el HDFS con los centros resultantes de la ejecución del algoritmo.

Para ejecutar el algoritmo de agrupamiento Canopy se empleará el comando presentado en el bloque de código 6.3, donde se invoca al método Canopy, y se indican como parámetros *-i*, el cual corresponde al directorio que contiene los vectores generados anteriormente, *-o* permite indicar el directorio en donde se almacenará la salida, y es justamente en aquí donde se generarán los centros que se emplearán posteriormente en el algoritmo de agrupamiento K-Means con lógica difusa. El parámetro *-dm* permite especificar el nombre de la clase con la que se medirá la distancia de los vectores hacia el centro de cada grupo, en este caso, será a través de la distancia euclidiana al cuadrado. Los parámetros *-t1* y *-t2* corresponden a dos umbrales de distancia para la ejecución del algoritmo. Para cada vector, si la distancia de éste hacia el centro es menor que *t1* entonces el vector es agregado al grupo. Por otro lado, si la distancia es menor que *t2*, entonces que vector es removido de dicho grupo, por ende T2 < T1, evitará que el algoritmo converja con alta probabilidad a los centros que fueron elegidos inicialmente. El último parámetro que se especifica es *-ow*, el cual, en caso de que existan resultados previos, estos serán sobrescritos.

```
mahout canopy -i Agrupamiento/tweets-vectores/tf-vectors -o Agrupamineto/
    tweets-vectores/tweets-canopy-centros -dm org.apache.mahout.common.
    distance.CosineDistanceMeasure -t1 2500 -t2 1000 -ow
```

Código 6.3: Algoritmo de agrupamiento Canopy en Mahout.

En la figura A.28 se muestra el contenido del directorio de salida especificado en el HDFS, en donde se aprecia que a diferencia de la ilustración A.27, se ha creado el directorio *tweets-canopy-centros*, el cual será empleado para la ejecución del algoritmo de agrupamiento de interés en esta sección.

Teniendo listos los centros iniciales es viable comenzar a ejecutar el algoritmo de agrupamiento K-Means con lógica difusa, esto se realiza mediante el comando *fkmeans*. Este comando es ejecu-

tado a través del bloque de código 6.4.

```
mahout fkmeans -i Agrupamiento/tweets-vectores/tfidf-vectors -c Agrupamiento/
    tweets-centros -o tweets-grupos -dm org.apache.mahout.common.distance.
    CosineDistanceMeasure -cd 0.1 -ow -x 20 -k 10 -m 1.5
```

Código 6.4: Algoritmo de agrupamiento K-Means con lógica difusa en Mahout.

En dicho bloque de código se establecen como parámetros *-i*, el cual corresponde al directorio con los vectores que serán agrupados, el parámetro *-c* permite definir el directorio en donde se encuentran los centros iniciales con lo que se ejecutará el algoritmo, este directorio contiene los centros con el formato requerido por el comando. Por otro lado, el parámetro *-o* permite especificar el directorio en el que se almacenarán los grupos generados. A través del parámetro *-dm* se especifica el nombre de la clase con la que se medirá la distancia de los vectores hacia el centro de cada grupo, en este caso será a través de la distancia cuadrática euclidiana.

El parámetro *-cd* corresponde al valor de convergencia delta, el cual es uno de los criterios de convergencia del algoritmo y se emplea para comparar entre cada iteración la mejora del resultado actual con el inmediato anterior. El parámetro *-ow* permite que en caso de que existan resultados previos estos sean sobrescritos. Otro de los criterios de parada es el número de iteraciones, el cual se especifica mediante el parámetro *-x*. El número de centros con los que se ejecutará el algoritmo es especificado mediante el parámetro *-k*, este valor deberá de corresponder con el número de centros obtenido mediante el método de Canopy. Finalmente, se emplea el parámetro *-m* para indicar el coeficiente de difusión, dicho valor debe de ser mayor a 1.

Es importante retomar el hecho de que el algoritmo en cuestión implementado en Mahout puede terminar su ejecución con base en tres criterios de parada. El primero de ellos es cuando ya no existan variaciones en los centros de cada grupo entre cada iteración, es decir, los centros de los grupos de la iteración actual son los mismos que los de la iteración anterior. El segundo es el criterio de convergencia delta, el cual evalúa los resultados de la iteración actual y busca que sean mejores que los previos. El tercer y último criterio es el número máximo de iteraciones, el cual detiene la ejecución en caso de que ninguno de los dos primeros criterios se hayan cumplido.

En la figura A.29 se presenta la ejecución en consola del comando presentado en el bloque de código 6.4, donde se observa que como parte de la salida de la ejecución se imprimen los parámetros

con los que se está ejecutando el algoritmo.

Después de que ha terminado la ejecución del algoritmo, se ha creado un directorio en el HDFS. Dicho directorio contiene un directorio por cada uno de los grupos generados, tal como se aprecia e la figura 6.7. En cada uno de ellos se encuentran los vectores que fueron asignados a estos grupos.

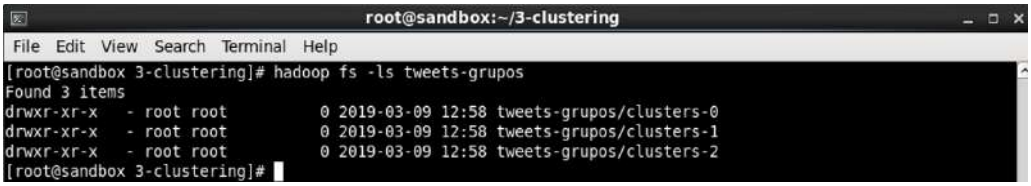


Figura 6.7: Salida de la ejecución del algoritmo de agrupamiento K-means con lógica difusa con Mahout.

6.3.2.2 Implementación de agrupamiento K-Means con lógica difusa con Python

Para la implementación del algoritmo de agrupamiento con el lenguaje Python se recurrirá a la biblioteca *scikit-learn*, la cual contiene diversos algoritmos comúnmente utilizados en aprendizaje de máquina. Para hacer uso de esta biblioteca se tendrá que realizar la instalación de la misma a través del comando *pip install -U scikit-learn*.

Mediante el bloque de código A.20 se lleva a cabo la ejecución del algoritmo en cuestión. El código comienza importando las bibliotecas necesarias para el algoritmo. El siguiente paso es cargar en el ambiente el corpus con el que se va a trabajar. Posteriormente, análogo a como se realizó en la implementación en Hadoop, se procede a obtener una representación métrica del corpus, para lo cual se generan vectores aplicando la técnica TF-IDF.

A partir de este corpus métrico, y teniendo definido el número de grupos con el que el algoritmo se ejecutará, se procede a definir los últimos parámetros del algoritmo, entre ellos el parámetro de difusión y el número máximo de iteraciones. De esta forma, se ejecuta el algoritmo de agrupamiento en cuestión.

Por último, el bloque de código finaliza con la identificación de los miembros que pertenecen a cada grupo, almacenando cada tweet en su correspondiente grupo en el archivo de salida. El proceso descrito anteriomente se presenta con mayor detalle en la sección A.3 del anexo A.

Para verificar que la ejecución del bloque de código A.20 no presentó problemas durante su ejecución se presenta la figura A.30, la cual contiene la salida en consola de la ejecución de dicho bloque de código, se ha agregado que imprima en consola a algunos de los miembros pertenecientes a cada uno de los grupos.

Después de verificar que el flujo de trabajo descrito previamente no ha presentado problemas durante su ejecución, y que además, los resultados son consistentes conforme a lo esperado, se concluye esta tarea analítica implementada con el lenguaje Python.

6.3.2.3 **Implementación de agrupamiento K-Means con lógica difusa con RapidMiner**

Para este ambiente, el corpus con el que se va a trabajar se encuentra almacenado como parte del proceso descriptivo ejecutado previamente (sección 6.3.1.3) en este ambiente.

Para la implementación del método de agrupamiento K-Means con lógica difusa se recurrirá a un módulo de la extensión *Information Selection*, después de tener instalada dicha extensión en el ambiente se procederá a diseñar el flujo de trabajo.

Como primer módulo a emplear se encuentra *retrieve*, el cual permitirá cargar el corpus. Posteriormente, de manera análoga a como se ha realizado en ambientes previos, se tendrá que realizar la vectorización del corpus a través de la técnica TF-IDF, lo cual será atendido por el módulo *Text-Vectorization*, y, en el cual deberá especificarse el atributo del corpus con el que trabajará. En este caso dicho atributo corresponde al del texto (*text*).

La salida del módulo de vectorización será redirigida hacia el módulo correspondiente al método de agrupamiento K-Means con lógica difusa, este módulo se llama *Fuzzy C-Means*. Entre los parámetros que son requeridos para la ejecución de dicho módulo se encuentran el número de grupos, el número máximo de iteraciones (como uno de los criterios de convergencia), el grado de difusión y el tipo de medición que se empleará (en este caso será Euclidiana).

El proceso descrito anteriormente se ilustra en la figura 6.8, se presentan todos los módulos descritos previamente.

Para determinar el número óptimo de grupos se buscó emplear el método de *elbow*, similar a como se realizó con Python (sección 6.3.2.2). Sin embargo, debido a que el módulo *Fuzzy C-Means* no cuenta con las métricas necesarias no fue posible emplear dicho método.

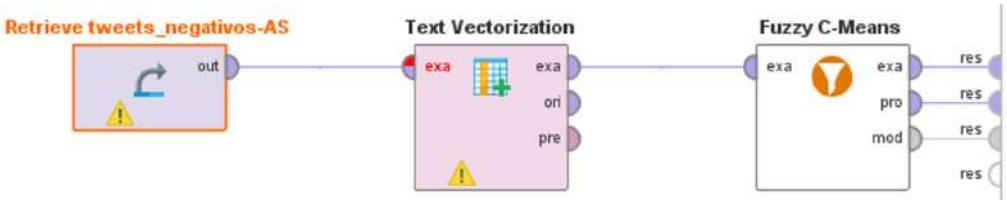


Figura 6.8: Diagrama de los componente involucrados para método de agrupamiento K-Means difuso con RapidMiner.

No obstante, se recurrió a las métricas comprendidas dentro de la herramienta y se encontró el módulo *performance*, el cual permite recurrir a la métrica de validación interna *Davies-Bouldin*.

Tal como se mencionó en la sección 3.2.2.2, este índice para cada grupo determina aquella relación máxima existente entre el la suma de la distancia promedio del centro del grupo a evaluar con respecto a sus vectores, más la distancia del centro que se está evaluando con respecto al centro de otro grupo divido entre la distancia de los centros de ambos grupos. En consecuencia, el número de grupos que minimiza esta métrica se toma como el óptimo.

Para implementar esta métrica de validación se modificó el diagrama presentado en la figura 6.8, añadiendo el módulo *performance* a la salida del módulo *Fuzzy C-Means*. Después de ejecutar el flujo de trabajo nuevamente se concluyó que esta métrica tampoco resultó ser de utilidad.

Debido a las características de los vectores, tienen una dimensionalidad tan grande que no es posible obtener un valor para este índice, esto se verifica con la salida de este módulo mostrada en la figura A.31.

Después de lo anterior, se optó por emplear el mismo número de grupos que se utilizó en las implementaciones previas de este algoritmo, así que el algoritmo de agrupamiento K-Means con lógica difusa será ejecutado para tres grupos. En la figura A.32 se presenta el resumen del modelo que se obtuvo después de haber ejecutado el flujo del diagrama 6.8. En dicha imagen, se presenta el número de vectores que fueron asignados a cada grupo.

La verificación de la salida del modelo que se ha ejecutado es el último paso de este proceso, por lo cual en la figura A.33 se presenta parte de la salida, en donde se aprecia que ha sido agregado un atributo correspondiente al grupo al que fue asignado el vector, el resto de los atributos corresponden a los tokens dentro del corpus.

6.4 Implementación de la evaluación de resultados.

De acuerdo con la metodología identificada en la sección 5.1, la siguiente fase a implementar corresponde a la evaluación de los resultados de los procesos analíticos que se han ejecutado en este capítulo.

Para implementar esta fase, de acuerdo con el diagrama de la arquitectura presentado en la figura 5.1, esta fase se desarrollará con ayuda de hojas de cálculo, así como indicadores e información que ofrecen las propias herramientas analíticas utilizadas en las fases 3 y 4 del diagrama.

El proceso de evaluación de resultados se aplicará a los procesos analíticos implementados en este capítulo.

En lo concerniente al análisis sentimental se empleará la técnica de la matriz de confusión (abordada en la sección 3.2.1.1), mientras que para el método de agrupamiento K-Means con lógica difusa se contempla emplear los índices de calidad así como la distribución espacial de los vectores buscando verificar que los grupos obtenidos estén compactos.

Siendo el análisis sentimental el primer proceso analítico a evaluar (debido a que este se implementó con tres entornos analíticos diferentes) se tendrá que realizar una evaluación para cada una de estas implementaciones.

Para obtener la matriz de confusión se reservó una parte del corpus. Esta selección se realizó previo a la implementación el proceso analítico de opinión. La selección de dicha parte del corpus está conformada por 1,577 instancias y se llevó a cabo mediante un muestreo aleatorio simple sin reemplazo.

Después de tener seleccionados los tweets destinados para la generación de la matriz, la siguiente tarea consistió en llevar a cabo una revisión manual de tal forma que cada tweet fue etiquetado, esto con el propósito de que sea posible comparar la etiqueta asignada por el proceso analítico contra la real (asignada mediante la revisión manual), y así poder generar un indicador del desempeño general que tuvieron cada uno de los modelos analíticos de clasificación implementados en la sección 6.3.1.

6.4.1 Evaluación de análisis sentimental implementado con Hadoop

Teniendo listo el corpus etiquetado como resultado del proceso 6.3.1.1 y teniendo etiquetado de manera manual el corpus para la evaluación, entonces es posible implementar la matriz de confusión para el análisis sentimental efectuado con Hadoop.

Para llenar la matriz se debe recorrer cada una de las etiquetas, comenzando por la etiqueta correspondiente al valor *«positivo»*. Ya que este es el valor real de la polaridad del tweet, se cuenta el número de instancias que fueron clasificadas correctamente con la misma etiqueta, y el valor de este conteo se registra dentro de la matriz en la intersección de la etiqueta *«positivo»* con *«positivo»*. En este caso se tuvo un total de 102 instancias clasificadas correctamente, tal como se aprecia en la figura 6.9.

Posteriormente, se realizó el conteo de todas aquellas instancias que debido a que su polaridad real es *«positivo»* fueron clasificadas como *«neutral»*, dando un total de 128 registros. Y finalmente, se dado que la polaridad de la publicación es *«positivo»*, cuantas instancias fueron clasificadas como *«negativo»*, obteniendo un total de 81 registros. Completando así el primer renglón de la matriz.

Procediendo de manera análoga a como se realizó para la etiqueta *«positivo»*, se realiza el conteo nuevamente, pero ahora partiendo del hecho de que la etiqueta original tiene el valor *«neutral»*, de tal forma que se obtuvieron 92 registros clasificados como *«positivo»*, 721 clasificados correctamente como *«neutral»* y 76 clasificados incorrectamente como *«negativo»*. El mismo procedimiento se realiza para cuando la etiqueta real corresponde a *«negativo»*, obteniendo los conteos que se presentan en la figura 6.9.

		Valor Predicho		
		Positivo	Neutral	Negativo
Valor Real	Positivo	102	128	81
	Neutral	92	721	76
	Negativo	45	47	285

Figura 6.9: Matriz de confusión para análisis sentimental implementado con Hadoop.

Después de tener lista la matriz de confusión, se procede a obtener indicadores para cuantificar el desempeño del modelo analítico implementado. El primer indicador es la **exactitud**, la cual se

expresa como el número de instancias que fueron clasificadas correctamente dividido entre el total de observaciones.

Contextualizando con las etiquetas que se están manejando, la exactitud se expresa de la siguiente forma:

$$Exactitud = \frac{(pos/pos) + (neu/neu) + (neg/neg)}{Total\ de\ observaciones}\ x\ 100\,\%$$

De tal forma que sustituyendo los valores de acuerdo a la matriz de la figura 6.9 se obtiene:

$$Exactitud = \frac{102 + 721 + 285}{1577}\ x\ 100\,\% = 70.26\,\%$$

El siguiente parámetro que se puede calcular a partir de los datos que han sido presentados en la matriz de confusión corresponde a la **tasa de error**, la cual indica el porcentaje de los datos que han sido clasificados de manera errónea. Contextualizando con este trabajo, la tasa de error se expresa como:

$$Tasa\ de\ error = \frac{(pos/neu) + (pos/neg) + (neu/pos) + (neu/neg) + (neg/pos) + (neg/neu)}{Total\ de\ observaciones}\ x\ 100\,\%$$

Sustituyendo las etiquetas por los valores correspondientes a los presentados en la matriz de confusión se llega al valor para la tasa de error siguiente:

$$Tasa\ de\ error = \frac{128 + 81 + 92 + 76 + 45 + 47}{1577}\ x\ 100\,\% = 29.74\,\%$$

Se observa que la tasa de error corresponde al complemento porcentual del valor de la exactitud obtenida, y derivado de que el valor obtenido en la exactitud es un valor bastante aceptable, se afirma que la tasa de error también lo es, ya que esto implica que de cada 10 instancias (tweets) que se clasifiquen, solo a tres se les asignará una etiqueta incorrecta.

El siguiente indicador a emplear corresponde a la **precisión** para cada una de las etiquetas. Este parámetro permite determinar el porcentaje que clasifica correctamente una etiqueta particular, por

ejemplo, para la etiqueta *«negativo»* se expresa de la siguiente forma:

$$Precisión_{neg} = \frac{neg/neg}{(pos/neg)+(neg/neg)+(neu/neg)}\ x\ 100$$

De tal forma que al sustituir las etiquetas por los valores correspondientes en la matriz de confusión se obtiene:

$$Precisión_{neg} = \frac{285}{81+76+285}\ x\ 100 = 64.48\,\%$$

A partir de los valores de los indicadores que se han calculado para esta sección se puede concluir que el modelo descriptivo implementado con el ambiente de Hadoop ha presentado un desempeño no óptimo, pero es tolerable para los fines de este trabajo.

6.4.2 Evaluación de análisis sentimental implementado con Python

Al igual que se realizó con los resultados obtenidos con el ambiente de Hadoop, se procederá a generar la matriz de confusión para el análisis sentimental implementado mediante el lenguaje Python.

Para generar dicha matriz se emplea el bloque de código A.21. Teniendo en el ambiente los vectores con las polaridades (uno para los valores reales y otro para los predichos), basta con importar la biblioteca *Sklearn.Metrics*, en donde se encuentra el método *confusion_matrix*, el cual recibe como parámetros obligatorios los dos vectores mencionados previamente. Dichos vectores han sido leídos del resultado del proceso analítico en cuestión y del corpus etiquetado designado para esta sección, cuyos valores son almacenados en las variables *y_test* (para los valores reales) y *y_pred* (para las etiquetas generadas por el proceso analítico). Como parámetro opcional se pueden indicar las etiquetas, las cuales han sido definidas en la línea tres mediante la variable *labels*.

Este método se encarga de realizar el conteo para cada una las etiquetas, y manda a imprimir en consola la matriz que se ha calculado. La salida de la ejecución se presenta en la figura 6.10, se aprecian los valores obtenidos de la matriz.

		Valor Predicho		
		Positivo	Neutral	Negativo
Valor Real	Positivo	245	61	5
	Neutral	231	653	5
	Negativo	26	26	325

Figura 6.10: Matriz de confusión para análisis sentimental implementado con Python.

Adicional a la obtención de la matriz de confusión, se realiza la graficación de los resultados de la matriz de confusión, esto con el propósito de tener una mejor apreciación de los resultados. En la ilustración 6.11 se presenta dicha gráfica.

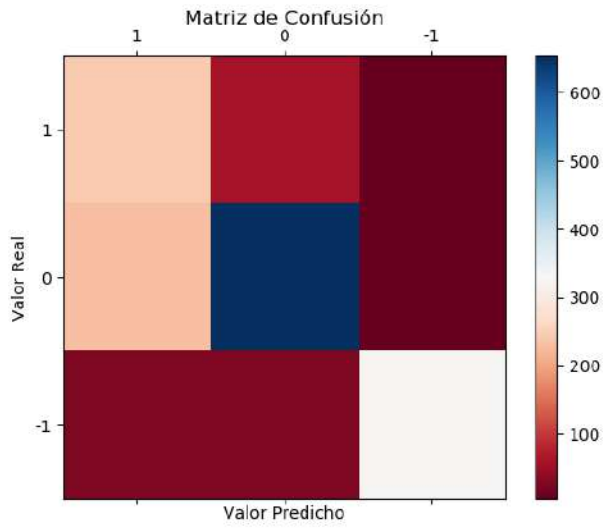


Figura 6.11: Gráfica de la matriz de confusión para análisis sentimental con Python.

En dicha ilustración, del lado derecho se presenta un gradiente de colores, indicándose que a mayor tendencia hacia el color azul se tienen valores mas grandes (dentro de todos los posibles de la matriz), por el contrario, a mayor tendencia al rojo se tendrán los valores más pequeños de la matriz de confusión.

Idealmente, se esperaría que la diagonal principal tuviera tendencias hacia el azul, pero a pesar de no ser así, se verá que al calcular los parámetros para la evaluación del modelo se observará que el modelo es tolerantemente aceptable.

Como primer parámetro a calcular se encuentra la **exactitud**, el cual, como se ha mencionado previamente, se expresa como el número de instancias que fueron clasificadas correctamente dividido entre el total de observaciones.

De tal forma que sustituyendo los valores de acuerdo a la matriz de la figura 6.10 se obtiene:

$$Exactitud = \frac{245+653+325}{1577}\ x\ 100\,\% = 77.55\,\%$$

El siguiente parámetro que se puede calcular a partir de los datos que han sido presentados en la matriz de confusión corresponde a la **tasa de error**, la cual indica el porcentaje de los datos que han sido clasificados de manera errónea. Empleando la misma expresión empleada en la sección 6.4.1, y sustituyendo los valores correspondientes a los presentados en la matriz de confusión se llega al valor para la tasa de error siguiente:

$$Tasa\ de\ error = \frac{61+5+231+5+26+26}{1577}\ x\ 100\,\% = 22.45\,\%$$

Como se ha mencionado, la tasa de error y la exactitud son valores relacionados, en consecuencia se puede decir que la tasa de error resultó ser menor que la obtenida mediante el análisis sentimental ejecutado con Hadoop, ya que con este clasificador por cada 10 instancias (tweets) que se analicen, únicamente dos serán clasificadas de manera incorrecta, lo cual posiciona a este clasificador por encima del implementado con Hadoop.

El siguiente indicador a emplear corresponde a la **precisión** (para las tres etiquetas), este parámetro permite determinar el porcentaje que clasifica correctamente cuando se asigna una etiqueta en particular. Sustituyendo los valores correspondientes en la matriz de confusión se obtienen las precisiones siguientes:

$$Precisión_{pos} = \frac{245}{245+231+26}\ x\ 100 = 48.80\,\%$$

$$Precisión_{neu} = \frac{653}{61+653+26}\ x\ 100 = 88.24\,\%$$

$$Precisión_{neg} = \frac{325}{5+5+325} \, x \, 100 = 97.01\,\%$$

A partir de los resultados anteriores se puede concluir que este modelo presenta un desempeño casi perfecto para la clasificación de tweets etiquetados como *«negativo»*, lo cual para el proceso analítico posterior es de gran importancia. Por otro lado, para las etiqueta *«positivo»* tiene una precisión muy deficiente, sin embargo, la clasificación de esta etiqueta carece de interés para el siguiente proceso analítico, concluyendo así que los parámetros de este modelo son aceptables.

6.4.3 Evaluación de análisis sentimental implementado con RapidMiner

Procediendo de manera análoga a como se realizó para las dos implementaciónes previas del análisis sentimental, a partir de los resultados obtenidos mediante análisis sentimental ejecutado con RapidMiner y contando con el corpus designado para validación, se procede a generar la matriz de confusión correspondiente.

La matriz de confusión obtenida se presenta en la figura 6.12, tal como puede apreciarse. Como primera aproximación se deduce que el proceso analítico evaluado es bastante estricto en la clasificación, pues la mayoría de los valores se encuentran en la columna central, esto derivado de que la mayor parte de los comentarios cuya etiqueta corresponde a *«positivo»* o *«negativo»* el análisis sentimental les asignó la etiqueta *«neutral»*.

		Valor Predicho		
		Positivo	Neutral	Negativo
Valor Real	Positivo	77	201	33
	Neutral	11	868	10
	Negativo	1	369	7

Figura 6.12: Matriz de confusión para análisis sentimental con RapidMiner.

A partir de los valores obtenido en la matriz anterior, se procede a calcular los parámetros de evaluación, comenzando con la **exactitud**, la cual se expresa como el número de instancias que fueron clasificadas correctamente dividido entre el total de observaciones.

De tal forma que sustituyendo los valores de acuerdo a la matriz de la figura 6.12 se obtiene:

$$Exactitud = \frac{77+868+7}{1577}\ x\ 100\,\% = 60.37\,\%$$

A partir del valor calculado en la exactitud se podría concluir que este proceso analítico es bueno, sin embargo, al analizar otros parámetros se podrá ver que la conclusión no es del todo correcta. Además, con base en la exactitud, el clasificador implementado con RapidMiner se posicionó por debajo de los otros dos clasificadores utilizados.

El siguiente parámetro que se puede calcular a partir de los datos que han sido presentados en la matriz de confusión corresponde a la **tasa de error**, la cual indica el porcentaje de los datos que han sido clasificados de manera errónea.

Empleando la misma expresión empleada en la sección 6.4.1, y sustituyendo los valores correspondientes a los presentados en la matriz de confusión se llega al valor para la tasa de error siguiente:

$$Tasa\ de\ error = \frac{201+33+11+10+1+369}{1577}\ x\ 100\,\% = 39.63\,\%$$

A partir de estos dos primeros indicadores se puede decir que el clasificador tiene un desempeño tolerable, pues de cada 10 instancias que clasifica a cuatro de ellas se les asignará una etiqueta incorrecta.

El siguiente indicador a emplear corresponde a la **precisión**, este parámetro permite determinar el porcentaje que clasifica correctamente para una etiqueta en particular. Sustituyendo los valores correspondientes en la matriz de confusión se obtienen las precisiones siguientes:

$$Precisión_{pos} = \frac{77}{77+11+1}\ x\ 100 = 86.52\,\%$$

$$Precisión_{neu} = \frac{868}{201+868+369}\ x\ 100 = 60.36\,\%$$

$$Precisión_{neg} = \frac{7}{33+10+7}\ x\ 100 = 14.00\,\%$$

De los valores para la precisión que se han calculado se puede concluir que este clasificador no es muy confiable de acuerdo con los objetivos de este trabajo, pues la clasificación de interés (negativa) tiene una precisión muy baja, el valor de 14.00 % indica que de cada 10 tweets negativos únicamente uno será clasificado correctamente.

En conclusión el alto valor obtenido en la exactitud aplica únicamente en la selección de aquellos individuos (tweets) cuya polaridad es positiva, ya que para las demás polaridades se tiene una precisión escasamente confiable. Además, este clasificador ha mostrado ser muy estricto para que un tweet sea etiquetado como negativo, teniendo en consecuencia un corpus bastante pequeño para el proceso de agrupamiento K-Means con lógica difusa, limitando desde este punto la efectividad del proceso de agrupamiento de acuerdo con los objetivos perseguidos en este trabajo.

6.4.4 **Evaluación del algoritmo de agrupamiento K-Means con lógica difusa implementado con Mahout**

Buscando algún factor que permita determinar el nivel de satisfacción del algoritmo de agrupamiento implementado en la sección 6.3.2.1 se llegó al hecho de que la biblioteca de Mahout no cuenta con índices de calidad, por mencionar algunos se encuentran el índice *Davies-Bouldin* o el *Coeficiente de Silhouette*, los cuales sí se encuentran disponibles en otras herramientas, tales como RapidMiner.

Esta ausencia de criterios por parte de la biblioteca de Mahout orilla a retomar la técnica expuesta en la sección 6.3.2. El método *El Codo* permite determinar el número óptimo de grupos con los que se debe de ejecutar el algoritmo de agrupamiento, funge como un indicador para garantizar que los grupos que se formen tendrán una variación pequeña en la inercia en comparación con el valor obtenido con un número diferente de grupos. Mediante este método se garantiza que los vectores de cada grupo se encuentran muy cercanos al centro del grupo.

6.4.5 Evaluación del algoritmo de agrupamiento K-Means con lógica difusa implementado con Python

Para determinar si el método de agrupamiento que se ha sido ejecutado en la sección 6.3.2.2 es aceptable, se retoma en primer lugar el método "*el codo*", el cual fue presentado en la sección 6.3.2, y a través del cual se ha logrado determinar el número óptimo de grupos para ejecución del algoritmo de agrupamiento K-Means con lógica difusa. Gracias a este método se determina el número de grupos con el cual la ejecución del algoritmo de agrupamiento genera una variación pequeña entre la inercia calculada en la instancia actual y la próxima (un grupo más que el número actual), de tal forma que se busca que los vectores pertenecientes a cada grupo estén lo más cercano al centro del grupo (minimización de la distancia intra-grupo). Esto facilitará la obtención de grupos los más homogéneamente posible.

Por otro lado, después de que se ha ejecutado el algoritmo de agrupamiento en cuestión, otro indicador a emplear es la visualización de la distribución espacial de los vectores agrupados. Para ello, después de que se han obtenido los vectores de características mediante el bloque de código A.20 se procede a realizar una reducción de dimensiones, de tal forma que se pueda obtener una gráfica de dos dimensiones.

Para llevar a cabo esta reducción de dimensiones se añaden las líneas presentadas en el bloque de código A.22 al bloque de código A.20, donde mediante el uso de Análisis de Componentes Principales (ACP) se describe el conjunto de vectores en términos de variables no correlacionadas. Este método persigue que se tenga la menor pérdida de información posible.

Las nuevas variables son combinaciones lineales de las anteriores y se van construyendo según el orden de importancia en cuanto a la variabilidad total que recogen de los datos.

La graficación de los vectores en las nuevas coordenadas, se presenta en la figura 6.13.

En la figura 6.13 se tiene una apreciación inmediata de la formación de tres grupos que han sido marcados mediante óvalos. Donde uno de ellos contiene un número mayor de vectores en comparación con los otros grupos, lo cual concuerda con los resultados presentados previamente en la figura A.32.

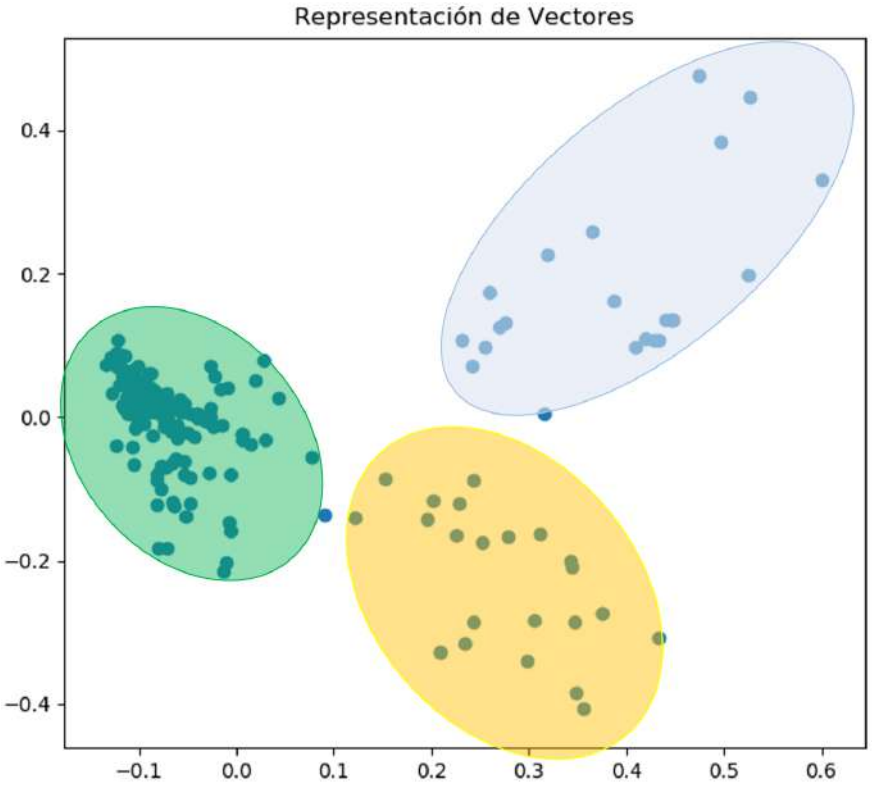


Figura 6.13: Visualización de vectores mediante ACP.

De los dos grupos que no se encuentran altamente aglutinados puede observarse que por más que se busque alguna forma de agrupar a los vectores de estos grupos, de tal modo que los vectores se encuentren más cercanos del centro, no es posible encontrar una solución factible ya que los vectores se encuentran muy dispersos. Así que puede deducirse que el algoritmo de agrupamiento que se ha ejecutado en la sección 6.3.2.2 cuenta con los criterios necesarios para que su resultado sea considerado aceptable.

6.4.6 **Evaluación del algoritmo de agrupamiento K-Means con lógica difusa implementado con RapidMiner**

Retomando los criterios expuestos en la sección 6.3.2.3, a través de los cuales se logró determinar el número óptimo de grupos para la ejecución método de agrupamiento en RapidMiner, se trató de determinar el índice *Davies-Bouldin*. Sin embargo debido a la particularidades del corpus no es

posible determinar un valor que permita establecer un criterio de aceptación.

Buscando emplear otro criterio, se empleó el módulo *performance IS (clustering)*, el cual utiliza un conjunto de vectores y el conjunto de centros a evaluar.

Este módulo permite conocer el valor de la distancia intra-grupo, mediante la cual se puede tener una percepción de qué tan distantes se encuentran los vectores de cada grupo respecto de su centro.

En este caso el conjunto de vectores corresponde al corpus que fungió como entrada para el proceso presentado en la figura 6.8, mientras que para el grupo de centros se emplearon aquellos generados después de la terminación de la ejecución del proceso representado en la misma figura.

El resultado de la ejecución de este proceso al igual que la determinación del índice *Davies-Bouldin*, ha sido de escasa utilidad, ya que se obtuvo el valor de *"unknown"* para este parámetro (tal como se ilustra en la figura A.34), asumiendo que esto se deriva de la limitada capacidad de procesamiento de la herramienta analítica.

Lo anterior no permite establecer un criterio de aceptación para el algoritmo de agrupamiento K-Means con lógica difusa ejecutado con RapidMiner, quedando como único parámetro aceptable la determinación del número de grupos realizada en la sección 6.3.2.

A continuación, se presenta en la figura A.35 la distribución de los grupos y sus densidades.

Análogo a los resultados obtenidos en los otros ambientes, un grupo es el que tiene la mayor concentración de vectores, mientras que los dos restantes parecieran tener una carga aproximadamente balanceada.

6.5 Implementación de la visualización de resultados.

La sexta y última fase de la metodología de desarrollo presentada en la sección 5.1 corresponde a la visualización de resultados, y de acuerdo con el diagrama presentado en la figura 5.1 es en esta sección donde mediante el empleo de gráficos y tablas se obtendrá una presentación de los resultados. De tal forma que sean comprensibles con mayor facilidad.

6.5.1 Distribución de los sentimientos en el corpus

De acuerdo con el flujo de trabajo que se implementó en la sección 6.3, los primeros resultados obtenidos corresponden a la salida del proceso analítico de opinión implementado en la sección 6.3.1. Ahí se obtuvo una etiqueta a partir de la polaridad contextual para cada una de las publicaciones (tweets) del corpus. Esta clasificación se implementó mediante tres herramientas tecnológicas diferentes: la primera de ellas fue Hadoop. A partir de los resultados generados con esta implementación y con ayuda de una hoja de cálculo se contabilizó el número de publicaciones para cada una de las etiquetas empleadas, de tal forma que los datos obtenidos de dicho conteo se resumen en la tabla 6.1.

Distribución de opiniones con Hadoop		
Polaridad	Número de instancias	Procentaje correspondiente
Positivo	4,387	79.44 %
Neutral	137	2.48 %
Negativo	998	18.07 %
Total 5,522 instancias		

Tabla 6.1: Polaridades obtenidas mediante análisis sentimental implementado con Hadoop.

Los datos de la tabla anterior permitieron generar la gráfica presentada en la figura 6.14, en la cual se aprecia la distribución de las polaridades obtenidas.

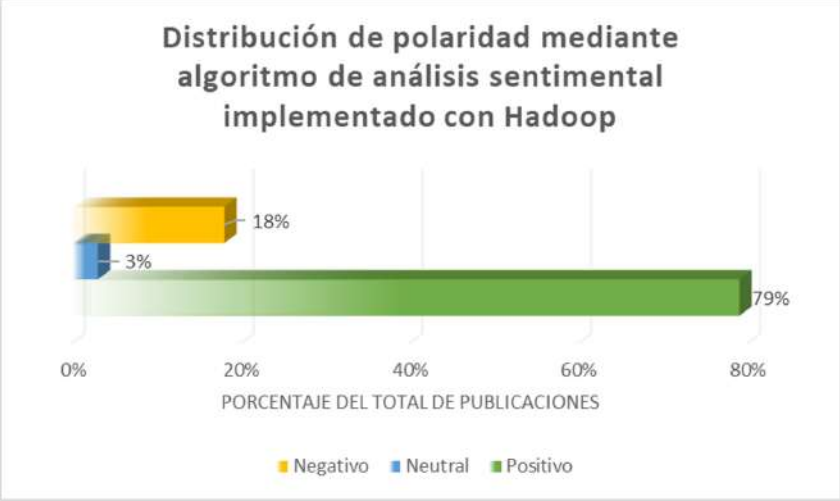


Figura 6.14: Distribución de polaridad mediante algoritmo de análisis sentimental implementado con Hadoop.

La figura 6.14 permite tener una percepción de la distribución de las opiniones analizadas mediante Hadoop, logrando destacar una gran ventaja por parte de los comentarios cuya polaridad es positiva y donde prácticamente el 79 % de las publicaciones se encuentran bajo esta categoría.

La siguiente categoría predominante corresponde a las publicaciones negativas, siendo que al 18 % del corpus se le asignó esta etiqueta. Por último, el 3 % de las publicaciones tuvo una etiqueta neutral, lo cual de acuerdo con el tipo de análisis implementado con Hadoop hace sentido, ya que la etiqueta de *neutral* es asignada únicamente a a aquellas publicaciones que la suma de las ponderaciones de las palabras contenidas es igual a cero, o bien, no se encontró ninguna de las palabras contenidas en la publicación dentro del diccionario sentimental empleado.

Realizando el mismo proceso para los resultados que se obtuvieron con el análisis sentimental implementado con python se presenta la tabla 6.2, la cual contiene los valores resultantes del conteo del número de opiniones para cada una de las etiquetas.

Distribución de opiniones con python		
Polaridad	Número de instancias	Procentaje correspondiente
Positivo	1,555	28.16 %
Neutral	3,712	67.22 %
Negativo	255	4.61 %
Total 5,522 instancias		

Tabla 6.2: Polaridades obtenidas mediante análisis sentimental implementado con python.

A partir de lo datos presentados en la tabla 6.2, se presenta en la figura 6.15 un gráfico mediante el cual se aprecia de manera inmediata que mediante el análisis sentimental implementado con python predomina la polaridad neutral con el 67.22 % de los tweets, seguida por la polaridad positiva con el 28.16 %, dejando la polaridad negativa en tercer lugar con apenas el 4.61 % del corpus.

A partir de esta gráfica se puede concluir que el proceso analítico (análisis sentimental) implementado con python podría limitar el proceso de agrupamiento, pues el número publicaciones que se involucrarán en proceso de agrupamiento es muy pequeño en comparación con las otras implementaciones. Se tendrán que analizar los resultados obtenidos a través del algoritmo de agrupamiento K-Means con lógica difusa para verificar si este supuesto es correcto.

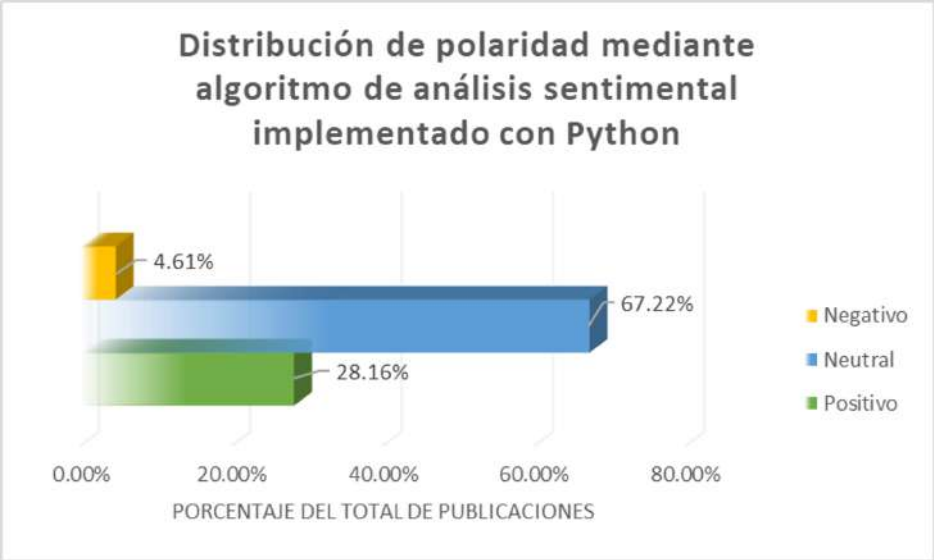


Figura 6.15: Distribución de polaridad mediante algoritmo de análisis sentimental implementado con python.

Después de haber visualizado la distribución de las opiniones clasificadas mediante los análisis sentimental con Hadoop y con python, resta realizar el mismo procedimiento con los resultados obtenidos a través de la última implementación la cual corresponde a RapidMiner.

El resumen de los resultados correspondientes a esta última implementación del análisis sentimental se presentan en la tabla 6.3. Es importante destacar que a través de la implementación del proceso analítico con RapidMiner se obtienen cinco etiquetas.

Distribución de opiniones con RapidMiner		
Polaridad	Número de instancias	Procentaje correspondiente
P+ (muy positivo)	282	5.10 %
P (positivo)	1,284	23.25 %
Neutral	73	1.32 %
N (negativo)	454	8.22 %
N+ (muy negativo)	108	1.96 %
None (ninguna)	3,321	60.15 %
Total 5,522 instancias		

Tabla 6.3: Polaridades obtenidas mediante análisis sentimental implementado con RapidMiner.

Los datos se la tabla anterior se presentan de manera gráfica en la figura 6.16, en donde resulta poco apreciable los valores de interés, es decir, aquellos valores correspondientes a las polaridades. La mayoría de las publicaciones no tuvieron asignación de alguna etiqueta, ocasionando que se descarte el 60 % del corpus.

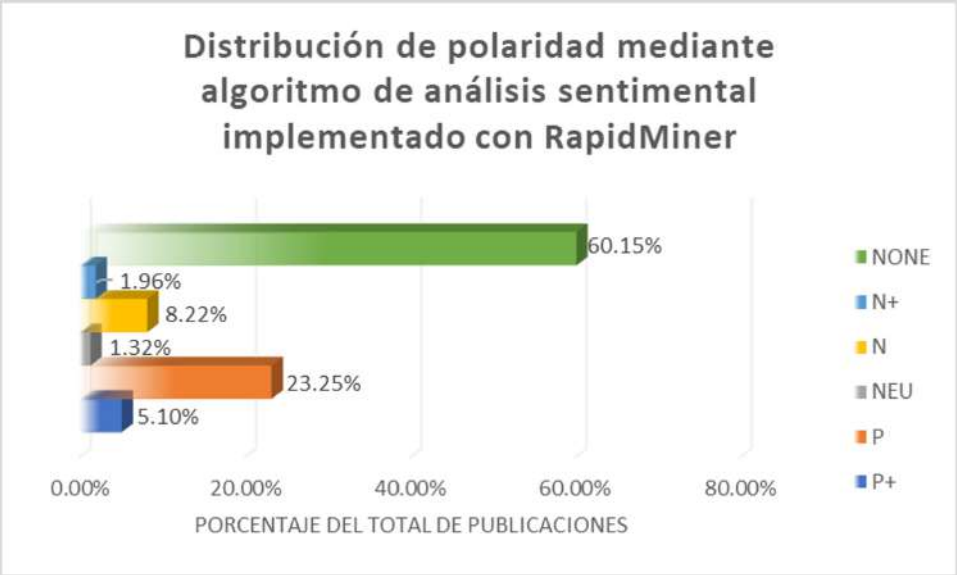


Figura 6.16: Distribución de polaridad mediante algoritmo de análisis sentimental implementado con RapidMiner.

Por lo anterior, en la tabla 6.4 se vuelven a acomodar los datos originales, pero ahora aplicando dos variantes. La primera consiste en que se descartan las opiniones que no expresan alguna polaridad, ya que estos carecen de utilidad de acuerdo con los fines perseguidos mediante este proceso analítico. Y por otro lado, como resulta igual de útil saber si una opinión es *positiva* o *muy positiva*, se realiza un reagrupamiento de los datos agrupando estas dos categorías en una sola. Esto mismo se realiza para las etiquetas *negativa* y *muy negativa*, de tal forma que los datos quedan de la forma presentada en la tabla 6.4

A partir de los datos presentados en la tabla 6.4 se genera un gráfico en la figura 6.17, mediante la cual se aprecia que de igual manera como acontece con la gráfica presentada en la figura 6.14 existe un predominio de la polaridad positiva seguida por la etiqueta de la polaridad negativa, dejando en última posición a la etiqueta neutral.

Distribución de opiniones con RapidMiner		
Polaridad	Número de instancias	Procentaje correspondiente
Positivo	1,566	71.14 %
Neutral	73	3.32 %
Negativo	562	25.53 %
Total 2,201 instancias		

Tabla 6.4: Reagrupamiento de polaridades obtenidas mediante análisis sentimental implementado con RapidMiner.

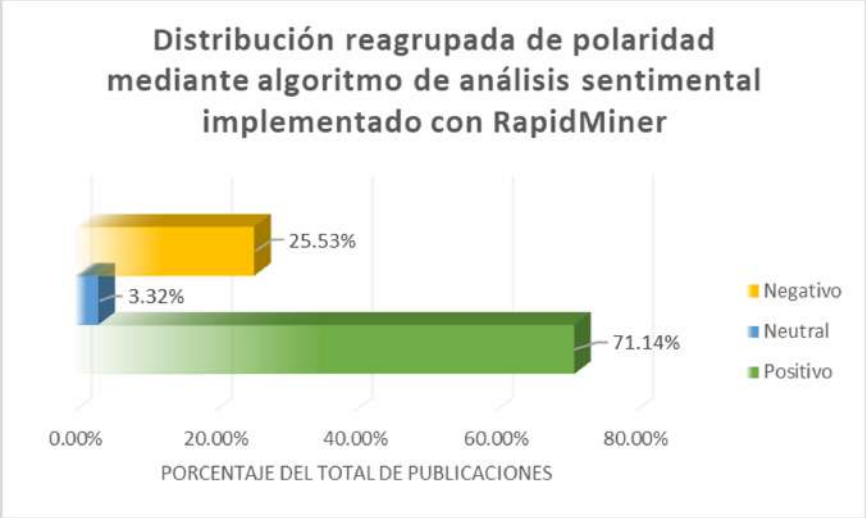


Figura 6.17: Reagrupamiento de la distribución de polaridad mediante algoritmo de análisis sentimental implementado con RapidMiner.

De acuerdo con las problemáticas presentadas en la sección 1.6, será de gran utilidad realizar un conteo para cada una de las polaridades respecto al periodo en el que fueron realizadas dichas publicaciones. Por lo anterior, respecto a los resultados obtenidos mediante el análisis sentimental implementado con Hadoop, en la tabla 6.5 se presenta el resumen del número de publicaciones para cada una de las polaridades respecto al mes en el que fueron realizadas dichas publicaciones.

Obteniendo una representación gráfica de los datos presentados en la tabla 6.5, se presenta en la figura 6.18 una gráfica en la cual se reflejan los resultados de la tabla en cuestión.

Distribución de opiniones por mes con Hadoop			
Mes	No. de positivos	No. de neutrales	No. de negativos
02/2018	406	12	91
03/2018	321	11	74
04/2018	115	8	26
05/2018	107	2	30
06/2018	120	3	46
07/2018	108	0	15
08/2018	102	3	18

Tabla 6.5: Comportamiento de las polaridades obtenidas con Hadoop en el tiempo.

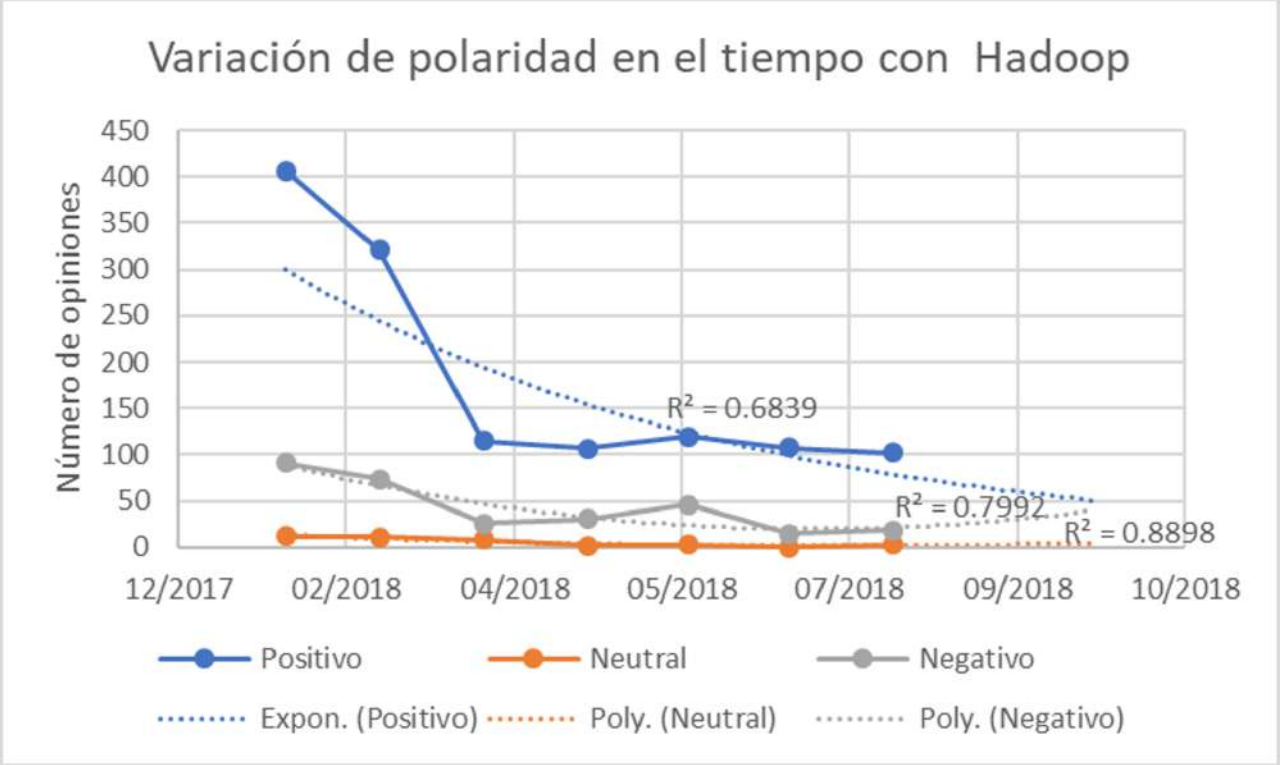


Figura 6.18: Comportamiento de las polaridades obtenidas con Hadoop en el tiempo.

Realizando el mismo resumen, pero ahora para los resultados obtenidos a partir del proceso analítico (análisis sentimental) implementado con python se genera la tabla 6.6.

Distribución de opiniones por mes con Python			
Mes	No. de positivos	No. de neutrales	No. de negativos
02/2018	506	1087	108
03/2018	392	586	46
04/2018	110	517	26
05/2018	120	405	21
06/2018	115	350	22
07/2018	125	304	15
08/2018	121	303	17

Tabla 6.6: Comportamiento de las polaridades obtenidas con python en el tiempo.

Buscando obtener una visualización mejor de los datos, se genera la gráfica presentada en la figura 6.19, en la cual se aprecia con mayor facilidad las variaciones del número de publicaciones por mes para cada una de las etiquetas asignadas por el proceso analítico en cuestión. Gracias a este gráfico se permite tener la percepción de que el número de publicaciones ha reducido drásticamente, lo cual probablemente sea consecuencia de la pérdida de popularidad.

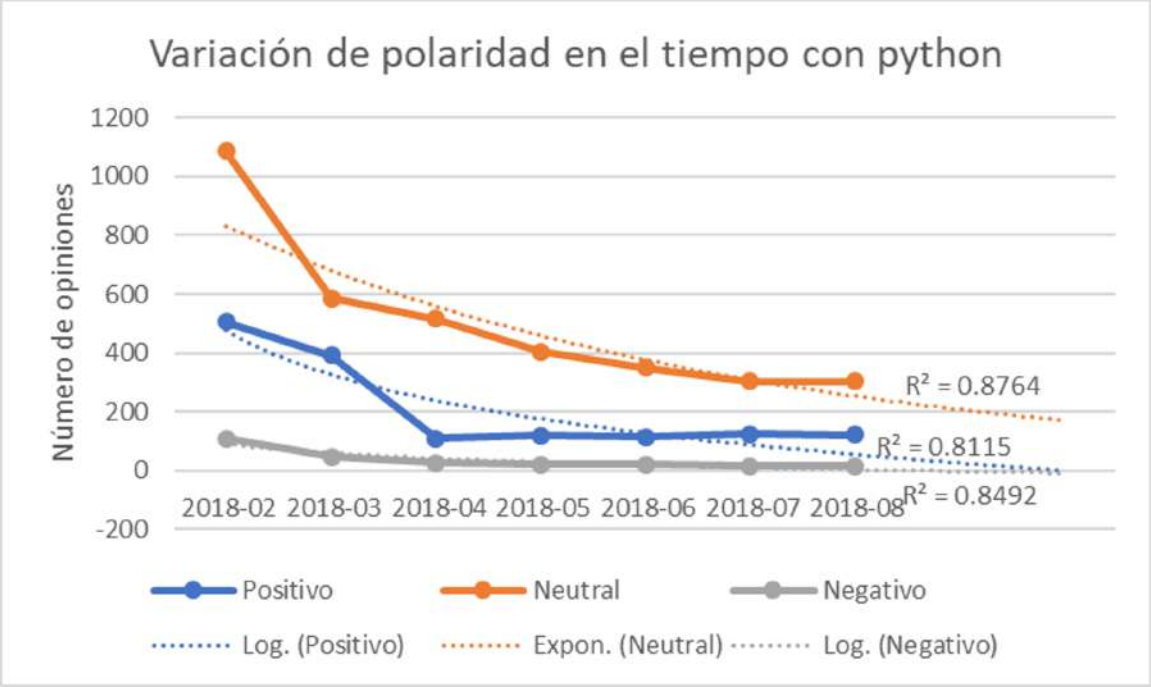


Figura 6.19: Comportamiento de las polaridades obtenidas con python en el tiempo.

Realizando este mismo proceso para los resultados obtenidos con la implementación del análisis sentimental con RapidMiner se genera la tabla 6.7, en la cual se resumen los resultados del proceso analítico en cuestión.

Distribución de opiniones por mes con RapidMiner			
Mes	No. de positivos	No. de neutrales	No. de negativos
02/2018	341	19	131
03/2018	144	5	52
04/2018	61	3	24
05/2018	64	1	18
06/2018	54	2	40
07/2018	65	6	22
08/2018	75	3	21

Tabla 6.7: Comportamiento de las polaridades obtenidas con RapidMiner en el tiempo.

A partir de los datos presentados en la tabla 6.7 se generó la gráfica presentada en la figura 6.20, a través de la cual se pueden apreciar las variaciones del número de publicaciones realizadas para cada una de las etiquetas en cuestión por periodo de publicación.

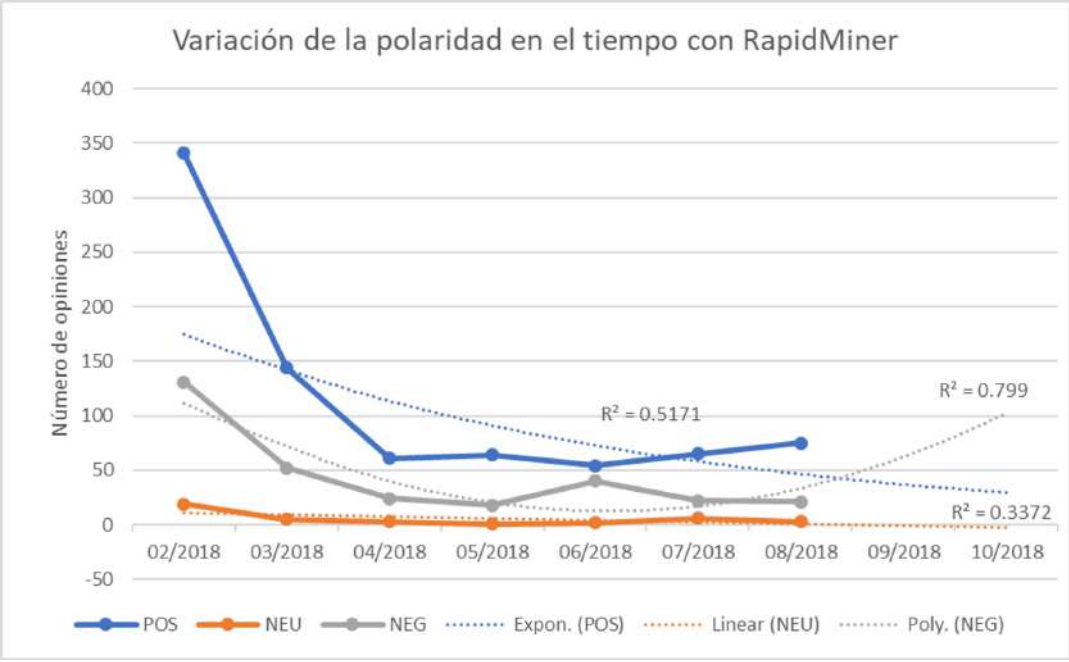


Figura 6.20: Comportamiento de las polaridades obtenidas con RapidMiner en el tiempo.

6.5.2 Identificación de características de los grupos

Después de concluir las representaciones visuales del análisis sentimental se procede a manipular los resultados obtenidos del siguiente proceso analítico implementado en la sección 6.3 (K-Means con lógica difusa). Para cada una de las ejecuciones del algoritmo con las diferentes herramientas tecnológicas empleadas se obtuvieron grupos con diferente número de miembros. Realizando una revisión de las características que tienen los miembros de los grupos, se proponen seis categorías en las que pueden clasificarse. Cada una de estas categorías permite tener una mejor percepción de las problemáticas expresadas por los usuarios de Twitter. Estas categorías y su correspondiente descripción se presentan en la tabla 6.8.

Abrev.	Categoría	Descripción
IS	Impacto social	Afectaciones derivados de manera directa de la implementación de la propuesta Amazon Go.
OL	Operación logística	Logística de operación dentro de la tienda.
ET	Empleo de tecnológica	Funcionamiento y tipo de tecnología empleada para la operación de la tienda.
P	Productos	Aspectos relacionados directamente con los productos: calidad, variedad y costo.
CS	Confianza en el servicio	El cliente no se siente completamente seguro con el servicio y/o la operación del mismo.
PI	Privacidad de la información	Preocupación por parte del cliente por privacidad y confidencialidad de su información.

Tabla 6.8: Categorías identificadas para la clasificación de los miembros de los grupos generados a través del algoritmo de agrupamiento K-Means con lógica difusa.

A partir de las categorías presentadas en la tabla 6.8, se realizó la revisión de los miembros pertenecientes a cada grupo, contabilizando el número de problemáticas y sugerencias identificadas con base a dichas categorías.

Realizando la identifición de características de los grupos resultantes de la ejecución del algoritmo K-Means con lógica difusa con Mahout se obtiene la tabla 6.9, en la cual se presenta el resultado del conteo por grupo de los tweets que pueden ser etiquetados dentro de alguna de las categorías propuestas, existe la posibilidad que algún tweet se le asigne más de una etiqueta.

Derivado de esta tabla se deduce que las problemáticas han sido agrupadas casi en su totalidad en

el grupo número 1; los tweets que pertenecen a los grupos 2 y 3 no pueden ser etiquetados bajo ninguna de las categorías de la tabla 6.8, sin embargo estos tweets poseen otras características, las cuales carecen de interés de acuerdo con los objetivos planteados en la sección 1.4.

Grupo	IS	OL	ET	Prod	Fal Conf	Priv
1	6	5	4	1	-	1
2	-	-	-	1	-	-
3	-	-	-	-	-	-

Tabla 6.9: Caracterización de los miembros los grupos obtenidos mediante algoritmo de agrupamiento K-Means con lógica difusa implementado con Mahout.

Procediendo de manera análoga con los grupos que han sido generados mediante la ejecución del algoritmo de agrupamiento K-Means con lógica difusa implementado con python se obtiene la tabla 6.10. Esta tabla permite identificar que los tweets que expresan alguna problemática perteneciente a alguna de las categorías propuestas se encuentran agrupados perfectamente en el grupo número 3. Este grupo es el de principal interés para este trabajo debido a que estos tweets responderán parte de las problemáticas presentadas en la sección 1.6.

Por otro lado, el grupo 1 está conformado por aquellos tweets que ofrecen alguna oferta o recomendación de productos de *Amazon Go*, mientras que el grupo 2 contiene aquellos tweets cuya opinión externa que *Amazon Go* no tiene garantizado el mercado; o bien, expresan su desaprobación a la propuesta *Amazon Go* a causa del posible monopólico crecimiento de ésta. A pesar de que estas opiniones son contrarias tienen en común su desapobración para *Amazon Go*.

Grupo	IS	OL	ET	Prod	Fal Conf	Priv
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	9	7	5	4	3	1

Tabla 6.10: Caracterización de los miembros los grupos obtenidos mediante algoritmo de agrupamiento K-Means con lógica difusa implementado con Python.

Los grupos generados mediante la ejecución del algoritmo de agrupamiento K-Means con lógica difusa implementado con RapidMiner son los últimos que se caracterizarán. Procediendo de ma-

nera análoga con el conteo por grupo del número de tweets que pueden ser etiquetados con alguna de las categorías presentadas en la tabla 6.8 se generó la tabla 6.11.

Grupo	IS	OL	ET	Prod	Fal Conf	Priv
1	-	-	-	-	-	-
2	-	-	1	4	1	1
3	2	1	-	-	-	1

Tabla 6.11: Caracterización de los miembros los grupos obtenidos mediante algoritmo de agrupamiento K-Means con lógica difusa implementado con RapidMiner.

A partir de la tabla 6.11 se puede apreciar que la mayoría de los tweets que expresan externan alguna problemática de acuerdo con las categorías que se están manejando se encuentran en el grupo número 2. Sin embargo, este grupo no es el único que contiene este tipo de opiniones, pues en el grupo 3 también se han identificado algunas problemáticas. Analizando el resto de los tweets que pertenecen al grupo 3 se logran identificar tweets que en general expresan alguna referencia a lo novedosa que es la propuesta *Amazon Go*.

Por último, en el grupo 1 no se han identificado problemáticas de acuerdo con las categorías que se proponen. En este grupo se encuentran aquellas publicaciones que ofertan o promocionan los productos ofrecidos por *Amazon Go*.

6.5.3 **Identificación de quejas**

A partir del análisis realizado en la sección anterior (sección 6.5.2) se han identificado las quejas externadas por los clientes, a continuación se presentan dichas quejas para cada una de las categorías que se han propuesto en la tabla 6.8. Se han unificado las quejas obtenidas a través de las ejecuciones del algoritmo de agrupamiento K-Means con lógica difusa en los diferentes entornos empleados, esto debido a que al haber utilizado el mismo corpus las quejas obtenidas en algunos casos se repiten.

- Impacto social
    - Existe preocupación por el reemplazo de los empleados por herramientas tecnológicas.

    - *Amazon Go* afecta directamente a las tiendas pequeñas, acaparando prácticamente el mercado.

- Operación logística
    - A gran parte de los usuarios les toma mucho tiempo registrar su aplicación para ingresar a la tienda.
    - Las personas que no cuentan con la aplicación (no se encuentran registradas) no pueden comprar en la tienda.
- Empleo de tecnología
    - La tecnología utilizada presenta problemas en el reconocimiento de los artículos.
    - Se han identificado usuarios que demoran demasiado tiempo en la tienda y al abandonarla no adquirieron ningún producto.
- Productos
    - Los productos ingeridos por el usuario le causaron algún malestar.
    - El costo de los productos es muy elevado en comparación con otras tiendas.
- Privacidad de la información
    - La información biométrica de los usuarios ahora es propiedad de la empresa.

6.5.4 **Identificación de sugerencias**

Procediendo de manera análoga a la sección 6.5.3, se presentan a continuación para cada categoría las sugerencias que son externadas en el corpus. Estos comentarios fueron clasificados como negativos a partir del análisis sentimental implementado, pero lo que expresan no consiste exactamente en una queja, más bien es una opinión que debido al léxico que conforma el tweet se ha etiquetado como negativo, pero en realidad comparte una sugerencia o un aspecto que se *Amazon Go* puede mejorar.

- Operación logística
    - Personas con capacidades especiales encuentran poco viable el utilizar el servicio.
    - Cuidar aspectos relacionados con actos delictivos por parte de los usuarios, e.g. tratar de robar productos o bandalizar la tienda.

    - Modificar el horario de operación de la tienda, de tal forma que haya servicio los fines de semana.

- Productos
    - Falta variedad de productos, se considera que se trata de una tienda más de bocadillos.
    - Existe oportunidad de ofrecer otro tipo de productos, los cuales se venden afuera de la tienda, tales como un aderezo.
- Confianza en el servicio
    - El cliente desea recibir y verificar su recibo de compra antes de dejar la tienda, esto por posibles aclaraciones.

6.5.5 Comparación de tiempos de ejecución

Buscando obtener un parámetro de comparación entre las tres herramientas tecnológicas que se emplearon en este trabajo, debido a que los procesos analíticos implementados se implementaron en las tres herramientas, resulta factible llevar a cabo una comparación de los tiempos de ejecución de cada una de ellas. Es por ello que a continuación se presenta la obtención de tiempos de ejecución que tomó a cada una de las implementaciones.

Comenzando con Hadoop, debido a que todo se manejó como scripts resulta sencillo calcular el tiempo de ejecución de los procesos, basta con emplear el comando *time*, el cual pertenece a la consola de comando de CentOS. Para ejecutar este comando basta anteponerlo al comando o proceso del cual se desea conocer el tiempo de ejecución.

La salida de este comando devuelve tres tiempos: el primero de ellos corresponde al tiempo del sistema (derivado de la ejecución de procesos y subrutinas de kernel); el segundo, corresponde al tiempo de usuario (generado a través del procesamiento por parte del usuario) y el tercero es el tiempo real.

El primer cálculo de tiempos corresponde a las tareas involucradas para llevar a cabo la limpieza de datos, para lo cual en la figura A.36 se presenta el tiempo que tomó ejecutar los procesos correspondientes a esta sección.

Posteriormente, el siguiente tiempo a calcular corresponde a la ejecución del análisis sentimental implementado con Hadoop, para el cual en la figura A.37 se presenta la salida del comando *time* aplicado al script correspondiente.

Por último, queda por calcular el tiempo de ejecución del algoritmo K-Means con lógica difusa con Mahout, para lo cual en la figura A.38 se presenta el tiempo de ejecución de este proceso.

Después de obtener los tiempos de ejecución de los procesos implementados en Hadoop, se realizará el mismo procedimiento pero ahora se calculará el tiempo de ejecución a los procesos ejecutados con Python. Para llevar a cabo esto se empleará el bloque de código A.23, en el cual lo que se hace se importar a las biblioteca *os* y *time*. La primera permitirá ejecutar los procesos mientras que a través de la segunda se obtendrá el tiempo.

La ejecución del bloque de código se ilustra mediante la figura A.39, en la cual a través de los mensajes de consola se indican los tiempos de ejecución para cada uno de los procesos implementados con python.

Finalmente, para los procesos ejecutados con RapidMiner no es necesario realizar ningún proceso o cálculo adicional, ya que la propia herramienta analítica cuenta con un gestor de tiempo, mediante el cual para cada proceso que se ejecuta dentro de RapidMiner se almacena su tiempo de ejecución. Por lo anterior, únicamente se recuperaron los datos proporcionados por la propia herramienta.

A continuación se presenta la figura 6.21, en la cual se presenta un resumen de los tiempos de ejecución para las implementaciones de los diversos procesos ejecutados en este trabajo.

Herramienta Analítica	Preparación de los datos	Análisis Sentimental	K-Means difuso	Tiempo Total
Hadoop	00:13:21	00:10:07	00:12:43	0:36:11
Python	00:00:29	00:00:03	00:00:03	0:00:35
RapidMiner	00:00:02	01:07:38	00:00:01	1:07:41

Figura 6.21: Comparación de tiempos de ejecución del proceso analítico con las diferentes herramientas analíticas utilizadas.

6.6 Conclusiones del capítulo

A través de este capítulo se implementaron las etapas presentadas de la metodología de desarrollo identificada en la sección 5.1.1. Partiendo de la adquisición de los datos en donde mediante la API de Twitter se logró obtener el corpus necesario y se lograron conocer los servicios que ofrece dicha API, se realizó la configuración y adecuación necesaria para poder utilizar dichos servicios.

Por otro lado, se implementaron las fases analíticas presentadas en la metodología desarrollo con tres herramientas analíticas diferentes, se conocieron las ventajas y dificultades presentes en cada una de ellas. A pesar de que cada una de estas herramientas tiene sus singularidades para la implementación de los procesos, se logró implementar de manera satisfactoria cada una de las tareas requeridas para cumplir con los objetivos de este trabajo. Comenzando por la preparación de los datos, en donde mediante procesos específicos se limpió y dió el formato requerido al corpus para que pudiera ser procesado por cada una de las herramientas analíticas.

Después de haber concluido las tareas de limpieza se implementó el primer proceso analítico (análisis sentimental), se usaron las bibliotecas y operadores ofrecidos por cada herramienta analítica de tal modo que en cada una de estas se obtuvo una clasificación del corpus; y, tal como se observó en este capítulo, cada implementación del análisis sentimental condujo resultados diferentes a pesar de trabajar con el mismo corpus.

A partir de los resultados del primer proceso analítico se segmentó el corpus, de tal forma que fue viable la implementación del algoritmo de agrupamiento K-Means con lógica difusa con cada herramienta analítica, obteniendo formaciones de grupos diferentes con cada implementación. A través de esta implementación se afrontaron los aspectos que debieron ser cubiertos para poder analizar el corpus, tales como formatos y pasar de una representación textual a una numérica del corpus.

Posteriormente, se evaluó a cada uno de los procesos analíticos implementados, llegando a un criterio de aceptación con base en los resultados obtenidos por cada proceso. Finalmente, se procedió a la visualización de los resultados, en donde estos fueron adecuados para obtener una mejor comprensión de los mismos y, con ello, el descubrir conocimiento.

CAPÍTULO

7

Análisis de resultados y conclusiones

A través de todo el proceso implementado, desde la investigación y recopilación presentada en el marco teórico, pasando por el estudio de algunas de las metodologías de desarrollo de aplicaciones analíticas Big Data (las cuales permitieron identificar y elegir los pasos de una metodología que se siguió a lo largo de este trabajo), es que se han logrado implementar los procesos analíticos de interés con el propósito de dar respuestas a las problemáticas planteadas en la sección 1.6 atendiendo así a las problemáticas externadas por los usuarios de *Amazon Go* en Twitter.

En la primera sección de este capitulo se analizaran los resultados de forma tal que se pueda dar respuesta a las problemáticas que se plantearon al inicio de este trabajo. Posteriormente, se analizarán las evaluaciones de los modelos obtenidos, buscando establecer criterios que permitan establecer un punto de comparación y así poder externar la preferencia por un modelo. Adicional a esta elección, también se compararán los desempeños de las tres herramientas analíticas utilizadas para la ejecución de los procesos, de tal forma que se podrá determinar la más adecuada para trabajar con procesos analíticos de Big Data.

7.1 Solución a las preguntas planteadas

Buscando dar respuesta a las preguntas propuestas en la sección 1.6 se retoman a continuación cada una de estas cuestiones y a partir de los resultados presentados en la sección 6.5 se buscará dar respuesta a dichas preguntas. Es importante destacar que la respuesta a estas preguntas está basada en la información que se obtuvo a partir de las publicaciones en Twitter. Sin embargo, para poder tomar decisiones en el negocio sería conveniente considerar de manera adicional otras fuentes de información que integren a usuarios de *Amazon Go* que no hacen uso de la red social en cuestión.

- **¿Con base en el comportamiento de las opiniones de los clientes, existe viabilidad de abrir nuevas sucursales?**
  Analizando las distribuciones de las polaridades obtenidas a través del análisis sentimental implementado con Hadoop (figura 6.14), se determina que existe un alto contenido favorable dentro de Twitter para esta nueva propuesta, pues la cantidad de opiniones positivas es cuatro veces mayor que el número de expresiones negativas. Esto se ratifica con los resultados obtenidos a través del análisis sentimental implementado con RapidMiner, en la figura 6.16 se observa que perdomina la polaridad positiva con un número de instancias casi tres veces mayor que el de publicaciones etiquetadas como negativas.
  En caso de que se tomen en consideración los resultados de las exactitudes obtenidas para cada uno de estos modelos en la sección 6.4.2 se podría tener en consideración que el clasificador empleado con RapidMiner ofrece la exactitud de 60.37 %, la cual es la menor en comparación con los otros dos clasificadores. Sin embargo, también debemos de considerar que este mismo clasificador tiene una alta precisión para las predicciones positivas, por lo cual se tiene confianza en la alta popularidad de las publicaciones positivas de acuerdo con los resultados obtenidos a través de este clasificador.
  Finalmente, a pesar de que en la gráfica 6.15 correspondiente a los resultados obtenidos a través del clasificador implementado con python no predomina la etiqueta correspondiente a las clasificaciones positivas, al enforcar la atención únicamente a las polaridades positivas y negativas se confirma que existe un predominio por parte de los comentarios con polaridad positiva, siendo la cantidad de comentarios positivos más de cinco veces mayor que la cantidad de publicaciones negativas. Hay que hacer énfasis en que este clasificador tiene una

exactitud del 77.55 %.

Por todo lo anterior, se puede concluir que sí existe viabilidad de abrir nuevas sucursales de *Amazon Go*. Hay gran aceptación por parte de los clientes, esto se debe principalmente a lo novedosa que resulta esta propuesta.

- **¿Los usuarios están satisfechos con la propuesta *Amazon Go*?**
  Retomando el análisis realizado en la pregunta anterior se deduce que existe una gran aceptación para la propuesta *Amazon Go* por parte de los usuarios. Sin embargo, como es de esperarse, existen opiniones que externan algún tipo de inconformidad, o bien, algún tipo de sugerencia. No obstante, esto representa una oportunidad para que el grupo comercial pueda mejorar el servicio ofrecido por *Amazon Go* buscando en consecuencia atraer más clientes.
- **¿Qué tipo de quejas son las que se presentan con mayor frecuencia?**
  A partir de la identificación de quejas presentada en la sección 6.5.3 y con base en los resultados presentados en las tablas 6.9, 6.10 y 6.11 se observa que la mayoría de las insatisfacciones se encuentran comprendidas dentro las categorías *Impacto Social* y *Operación Logística* con un total de 72 publicaciones con esta tendencia.
  Tomando la identificación de quejas presentada en la sección 6.5.3 y realizando la revisión de las quejas correspondientes a las dos categorías mencionadas anteriormente, se encontró que las quejas expresadas con mayor frecuencia son las que se listan a continuación:
  - Existe preocupación por el reemplazo de los empleados por herramientas tecnológicas.
  - *Amazon Go* afecta directamente a las tiendas pequeñas, acaparando prácticamente el mercado.
  - A gran parte de los usuarios les toma mucho tiempo registrar su aplicación para ingresar a la tienda.
  - Las personas que no cuentan con la aplicación (no se encuentran registradas) no pueden comprar en la tienda.
- **¿Cómo se comportará la popularidad de *Amazon Go* en Twitter?**
  Para dar respuesta a esta pregunta se analizarán las gráficas presentadas en la sección 6.5.1, las cuales corresponden a la distribución de las polaridades en el tiempo para cada modelo.
  Comenzando con la gráfica 6.18, la cual corresponde a los resultados obtenidos a través de la implementación del análisis sentimental con Hadoop, se observa que en general el número de publicaciones presenta decrecimiento cuya tendencia se asemeja a un decrecimiento exponencial. En la gráfica se presenta el comportamiento de cada una de las polaridades y,

para cada uno de estos comportamientos mediante algún tiempo de regresión, se ha logrado obtener la tendencia de su comportamiento futuro. Por ejemplo, para la distribución de los comentarios positivos (identificados mediante el color azul sólido) se observa que existe un decrecimiento en el número de publicaciones realizadas; y a través de la regresión exponencial, se obtiene el modelo (línea azul punteada) que permite predecir que el comportamiento de esta polaridad continuará decreciendo.

Para este modelo se obtuvo el valor para R^2 de 0.6839, se optó por dejar este modelo exponencial debido a que se busca evitar obtener un modelo sobre ajustado. Por otro lado, analizando las distribuciones de las polaridades neutrales (anaranjado sólido) y negativas (gris sólido) se ratifica el comportamiento en decrecimiento. Sin embargo, a partir de los modelos obtenidos mediante las regresiones polinomiales (anaranjado punteado para la polaridad neutral y gris punteado para la negativa) se observa que este decrecimiento nunca llega al valor de cero publicaciones.

Analizando de manera análoga el comportamiento de las polaridades obtenidas mediante el análisis sentimental implementado con python, se observa en la gráfica 6.19 que el decrecimiento del número de publicaciones prevalece y a partir de los modelos obtenidos mediante las regresiones (logarítmica para las polaridades positiva y negativa, y exponencial para la polaridad neutral) se predice, para los dos meses siguientes que el comportamiento decrecimiento prevalecerá.

Por último, realizando el mismo análisis con los resultados obtenidos a través del análisis sentimental implementado con RapidMiner, en la gráfica 6.20 se observa que el número de publicaciones decrece, y sus respectivas líneas de tendencia nos confirman este comportamiento para futuros meses.

En conclusión se ha visto que el comportamiento de la popularidad de *Amazon Go* va decayendo, sin embargo, esto no debe de tomarse con negatividad absoluta, ya que debe de comprenderse que recién que esta propuesta llegó a la ciudad de San Francisco el servicio se popularizó debido justamente a la novedad de este servicio. Considero que el aspecto en el que el grupo comercial debe de centrar su atención es que el número de publicaciones positivas prevalezca sobre el número de publicaciones negativas.

- **¿Cuáles son las sugerencias expresadas por los clientes?**

  A través de la identificación que sugerencias que se implementó en la sección 6.5.4, es que se puede dar respuesta a estas preguntas. De acuerdo a las categorías presentadas en la tabla

6.8, se identificaron 48 publicaciones que caen dentro de esta selección y se clasificaron las sugerencias expresadas por los usuarios. Las cuales se enlistan a continuación:

- Personas con capacidades especiales encuentran poco viable el utilizar el servicio.
- Cuidar aspectos relacionados con actos delictivos por parte de los usuarios, e.g. tratar de robar productos o bandalizar la tienda.
- Modificar el horario de operación de la tienda, de tal forma que haya servicio los fines de semana.
- Falta variedad de productos, se considera que se trata de una tienda más de aperitivos.
- Existe oportunidad de ofrecer otro tipo de productos, los cuales se venden afuera de la tienda, tales como un aderezo.
- El cliente desea recibir y verificar su recibo de compra antes de dejar la tienda, esto por posibles aclaraciones.

A partir de estas sugerencias detectadas el grupo comercial *Amazon* puede tomar ventaja, por ejemplo, atendiendo a la primera sugerencia de la lista y adecuando la tienda se puede incrementar el número de clientes.

Este efecto puede ser conseguido al atender la sugerencia que habla sobre la variedad de productos, porque un cliente se siente más cómodo si va a una tienda que tenga todos los productos que busca, en lugar de tener que ir visitando diversas tiendas para adquirir todos los productos que desea. Aunado a esto, la diversificación de productos y la novedad en la operación de la tienda permitirán que nuevos clientes se interesen por *Amazon Go*.

Por otro lado, el aspecto de la seguridad también es una sugerencia en la que el grupo comercial debe poner atención, ya que en caso de que no busque implementar mecanismos que controlen actos delictivos se comenzarán a ver pérdidas económicas.

En conclusión, a través de las sugerencias que han sido presentadas al buscar dar respuesta a esta pregunta se han logrado identificar aspectos de diversa índole, de tal forma que si éstos son atendido por el grupo comercial se puede obtener un beneficio para el servicio de *Amazon Go*. En primer lugar, se podrá conservar a los clientes, por otro lado, al ofrecer más productos se propicia que cada vez más usuario pongan su atención en la propuesta de este grupo comercial; y finalmente, a través de estas sugerencias se pude mejorar la seguridad para la propia empresa.

- **¿Cuál es la distribución de comentarios positivos y negativos?**

  Tal como se aprecia en las gráficas 6.14, 6.15 y 6.17, obtenidas a partir de las implementa-

ciones del análisis sentimental, se observa que en todos los casos prevalece predominio del número de publicaciones con polaridad positiva sobre el número de tweets con polaridad negativa, y tal como se analizó en la primera pregunta de esta sección, en cada una de las gráficas mencionadas se presenta una proporción bastante mayor por parte de las publicaciones positivas, lo cual es algo positivo para lo propuesta porque permite concebir que el servicio *Amazon Go* está siendo aceptado por los usuarios.

- **¿Cómo será el comportamiento de las opiniones de los clientes de *Amazon Go* en Twitter a futuro?**

  A partir de las gráficas 6.18, 6.19 y 6.20, las cuales corresponden a las distribuciones de las polaridades en un periodo de tiempo determinado, se observa que en cada una de estas gráficas se modela el comportamiento de cada polaridad. Para obtener el modelo de cada distribución se empleó una regresión, en algunos casos fue de tipo exponencial, en otros logarítmica y en otros polinomial. Estos modelos se han presentado a través de una línea punteada y el color que emplean tienen correspondencia con el la distribución modelada.

  Cada uno de estos modelos se encuentra acompañado por su valor correspondiente de R^2. Para cada distribución se ha buscado obtener un modelo que se ajuste a la distribución de los puntos sin llegar al sobreajuste del modelo, ya que en este caso se podría obtener una predicción que muy probablemente sea incorrecta.

  Con base en estos modelos obtenidos, en cada una de las gráficas presentadas se ha realizado una predicción para cada una de las polaridades hacia los dos siguientes meses; y, a través de las tres gráficas se observa que el comportamiento de las opiniones va en decaimiento. Sin embargo, las tendencias nos predicen que el número de publicaciones no llegará cero, que probablemente se mantendrá oscilando entre valores comprendidos entre cincuenta y cien publicaciones por periodo de observación.

- **¿Cuál será la popularidad de los tweets con contenido desfavorable para la empresa?**

  Buscando responder a esta pregunta nuevamente se recurre a las gráficas 6.18, 6.19 y 6.20, mediante las cuales se observa que el comportamiento de las publicaciones que han sido etiquetadas con contenido negativo es minoría en comparación con las otras dos categorías. Este comportamiento se preserva en las tres gráficas en cuestión.

  Analizando la tendencia que se presenta en cada una de las gráficas mencionadas, se puede predecir que el comportamiento de las publicaciones con contenido negativo continuará posicionándose por debajo de demás polaridades. Además, cabe destacar el hecho de que el

número de publicaciones en todos los periodos de tiempo son pequeños, y que con base en la tendencia el número de publicaciones negativas que se esperan por mes en promedio son veinte, mientras que para la polaridad positiva se esperan al rededor de cien publicaciones durante el mismo periodo.

- **¿Cuáles son los productos más populares dentro de las opiniones de *Amazon Go* en Twitter?**

  Para atender esta pregunta se trabajó con el corpus después de que este había sometido a la fase de preparación de los datos, realizándose la identificación de los términos con mayor frecuencia sobre todo el corpus. A partir de esta identificación se tuvo que realizar una revisión manual para filtrar aquellos términos que hacen referencia a un producto, de tal forma que se encontró que el producto más popular dentro del corpus corresponde a revistas apareciendo ochenta y un veces en el corpus, seguido por los términos sandwich, lunch y yogurth.

  A partir de esta identificación de productos se puede intuir que la mayoría de los clientes adquieren productos que consumen como un refrigerio, esto puede permitir al grupo comercial abarcar dos aspectos. El primero de ellos consiste en aumentar la variedad de los productos que ofrece buscando que los clientes adquieran algo más que un refrigerio, por ejemplo, comercializar productos de higiene personal en presentaciones pequeñas. Por otro lado, tal como se ha externado en la sugerencia identificadas en la sección 6.5.4, hay viabilidad de comercializar productos alimenticios diferentes, lo cual también favorecerá la atracción de nuevos clientes y, aunado a esto, la conservación de los viejos clientes al ofrecerles variedad de productos siempre tendrán una opción diferente de productos para consumir.

- **¿Es posible predecir un crecimiento en la difusión de los tweets con contenido positivo sobre *Amazon Go*?**

  Analizando las gráficas 6.18, 6.19 y 6.20, las cuales corresponden a los comportamientos de las distribuciones de las polaridades en el tiempo, se puede observar que el comportamiento de la polaridad positiva en cada una de estas gráficas tiene una tendencia negativa. Esto mismo se ratifica al revisar los modelos para cada una de las polaridades generadas mediante algún tipo de regresión, pues la predicción de cada modelo indica que el número de publicaciones negativas no aumentará.

  Sin embargo, como se ha mencionado anteriormente, el grupo comercial no debe de ver esto como algo duramente negativo, pues al comparar el comportamiento de las distribuciones

de las polaridades restantes se aprecia que el número de publicaciones positivas siempre es mayor que el número de publicaciones con contenido desfavorable y esta tendencia se preserva para futuros meses.

De esta forma se concluye la resolución de las preguntas que contribuyen al grupo comercial con algún tipo de conocimiento, a partir del cual el el grupo comercial puede tomar acciones correctivas que satisfagan las problemáticas identificadas en la sección 6.5.3 y además ofrecer un mejor servicio si considera las sugerencias.

Con base en el conocimiento que se ha presentado a través de la respuesta a estas preguntas el grupo comercial cuenta con soporte para realizar toma de decisiones. Por ejemplo, decidir si es viable abrir más sucursales o cómo cambiar los productos que ofrecen.

7.2 Elección de modelo con mejor desempeño

Para poder discernir el modelo que ha presentado el mejor desempeño (tanto para el análisis sentimental como para el algoritmo de agrupamiento K-Means con lógica difusa) se emplearán los resultados de las evaluaciones realizadas en la sección 6.5.

Realizando una recopilación de los valores para la exactitud, tasa de error y precisión para cada una de las etiquetas, se llega a la tabla presentada en la figura 7.1, en la cual se resumen todos los valores obtenidos mediante la evaluación de cada modelo.

A partir de la tabla anterior se puede observar que el modelo que presenta la mayor exactitud para la clasificación es el que se implementó con python, pero cabe destacar un aspecto muy importante que se observa para este modelo, y es justamente que su precisión para la clasificación es muy deficiente, pues asigna esta etiqueta de manera incorrecta la mitad de las ocasiones. Sin embargo, se observa que para los propósitos de este trabajo, el modelo ofrece una precisión muy alta para la clasificación de instancias negativas, garantizando que prácticamente todas las clasificaciones que realiza como negativas con correctas.

Analizando los otros dos modelos se observa que tienen una exactitud aceptable. Sin embargo, para el clasificador implementado con Hadoop se aprecia que acontece lo mismo que sucedió con el clasificador implementado con python, ambos tienen una precisión de clasificación de positivos

	Hadoop	Python	RapidMiner
Exactitud	70.26%	77.55%	60.37%
Tasa de error	29.74%	22.45%	39.63%
Precisión Positivos	42.68%	48.80%	86.52%
Precisión Neutrales	80.47%	88.24%	60.36%
Precisión Negativos	64.48%	97.01%	14.00%

Figura 7.1: Comparación del desempeño de las implementaciones del análisis sentimental.

muy baja, lo cual propiciará que la mitad de las publicaciones con esta etiqueta están asignados se incorrecta. Por otro lado, en los resultados obtenidos en la evaluación del clasificador implementado con RapidMiner se rescata el hecho de que este modelo se distingue de los otros dos por tener la mayor precisión en la clasificación de positivos; pero es muy deficiente para la clasificación de negativos.

Con base en lo anterior se puede decir que el modelo clasificador implementado con RapidMiner es el que ha presentado el peor desempeño y esto se ratifica con los resultados de la sección 6.5.2, pues en la tabla 6.11 se observa que a pesar de que se trabajó con el mismo corpus, a través de esta implementación se pudieron identificar muy pocas problemáticas en comparación con las tablas 6.10 y 6.9.

En conclusión, se puede deducir que de acuerdo con los objetivos que se persiguen con este trabajo, el mejor clasificador de opinión es el que se implementó con python, pues ha presentado tener la mejor exactitud de los tres modelos utilizados. Además, este modelo es superior a los otros dos en lo que respecta a la precisión para la clasificación de publicaciones con contenido negativo, pues la precisión que ofrece es muy cercana al 100 %. La elección de este clasificador se ratifica con los datos presentados en la tabla 6.10, pues gracias a éste clasificador se logró definir un corpus negativo, el cual cuando al ser sometido al algoritmo K-Means con lógica difusa se logró identificar el mayor número de problemáticas.

En lo concerniente a la elección del mejor modelo de agrupación se debe considerar que el desempeño de cada algoritmo se verá afectado directamente a partir del corpus que se emplee, pues tal como se observó con cada implementación del análisis sentimental en la sección 6.3.1 se obtuvieron diferentes corpus, para Hadoop el corpus negativo se conformó por 998 registros, mientras que el de RapidMiner tuvo 562 registros y python solo 255.

Antes de haber implementado el algoritmo de agrupamiento K-Means con lógica difusa con cada una de las herramientas analíticas se consideraba que el tamaño del corpus para la implementación con python limitaría se desempeño y, que en consecuencia, esta implementación identificaría muy pocas problemáticas. Sin embargo, al comparar el número de problemáticas identificadas por la implementación del algoritmo de agrupamiento con python (tabla 6.10) se concluye que se ha obtenido la mayor identificación de problemáticas (debido al corpus) y así mismo el mejor agrupamiento (todas las problemáticas se ubicaron en un único grupo) en comparación con los resultados presentados en las tablas 6.9 (correspondiente a la implementación con Mahout) y 6.11 (correspondiente a la implementación con RapidMiner).

En conclusión, gracias a la evaluación realizada en la sección 6.4 y con base en lo mencionado en párrafos anteriores, se puede concluir que el desempeño de la ejecución del algoritmo de agrupamiento K-Means con lógica difusa dependerá de manera significa del corpus que se emplee. Las tres implementaciones del algoritmo presentaron un desempeño bueno, pues como se analizó en la sección 6.5.2 cada una de las implementaciones de este algoritmo logró generar grupos cuyos miembros que los integraban tenían características en común, siendo aquí donde se identificaron grupos con características que carecían de interés para este trabajo, pero esta formación ratifica el buen desempeño por parte de los algoritmos.

Tomando como base la identificación de problemáticas, de acuerdo con las tablas 6.10, 6.9 y 6.11 se puede concluir que la implementación del algoritmo con el mejor desempeño fue el correspondiente a Python, pues logró agrupar perfectamente todas las problemáticas que contenía su corpus en un único grupo. Esto estuvo cerca de suceder con la implementación realizada con Mahout, pero no se logró debido a que se presentó una problemática en otro grupo. A pesar de ello, se concluye que esta implementación también ha sido muy satisfactoria.

7.3 Elección de la mejor herramienta analítica

En lo que respecta a la implementación de la adecuación de los datos, así como la implementación de los procesos analíticos en la sección 6.3 de este trabajo se observó que atendiendo las singularidades de cada herramienta se pudieron llevar a cabo todos los procesos requeridos para cumplir con los objetivos de esta investigación.

Can base en los resultados obtenidos correspondientes a los tiempos de ejecución que se presentaron en la figura 6.21 se observa que la herramienta que ocupó el mayor tiempo para ejecutar todos los procesos realizados fue RapidMiner, demorando más de una hora en concluir los procesos. Hadoop ocupa la segunda posición al demorar aproximadamente treinta y cinco minutos en realizar todo el procesamiento de los datos, mientras que python se posiciona en primer lugar al tener un tiempo de ejecución total menor a un minuto.

Por lo anterior puede posicionarse a python como la mejor herramienta para análisis de datos en este proyecto. Sin embargo, es necesario considerar que el tipo de análisis que se busca hacer es Big Data, en este trabajo se manipuló un corpus pequeño e invariable, de tal forma que la implementación de los procesos analíticos se facilitó. De este modo, con este trabajo se han establecido las bases para el análisis de Big Data, existiendo la posibilidad de escalar la arquitectura de este trabajo para que el análisis se implemente con cantidades masivas de datos, manejando así los grandes volúmenes de datos que son generados dentro de Twitter.

Otro aspecto a considerar son las limitaciones físicas en donde se implementó Hadoop, el cual está diseñado para trabajar con diversos nodos de forma paralela, debido a que la implementación en este trabajo se contó con un único nodo de procesamiento y que además esta instalación se realizó sobre una máquina virtual, el desempeño de Hadoop se vió limitado en gran medida. Además, la implementación con python se realizó de manera nativa, de tal forma que la implementación de los procesos de liempieza así como los analíticos aprovecharon todos los recursos de hardware y software de la máquina donde se ejecutaron, mientras que la implementación tuvo que operar sobre la capa de virtualización.

Por los aspectos mencionados anteriormente, a pesar de que python se ha posicionado como una herramienta con tiempos de respuesta cortos, se considera que esta herramienta no tendría la capacidad para manejar grandes volúmenes de datos en lapsos de tiempo tan pequeños (las publicacio-

nes dentro de la red social son muy cortos). En caso de que se deseara escalar la implementación con python se terminaría recurriendo a Hadoop como sistema de archivos. De tal forma que se concluye que el ecosistema de Hadoop es la mejor herramienta para atender las cuestiones analíticas de Big Data concernientes a este trabajo debido a su capacidad de escalamiento.

7.4 Observaciones finales y trabajo futuro

A través de la realización de este trabajo se ha logrado realizar un estudio de algunas de las metodologías de desarrollo para aplicaciones analíticas de Big Data, obtiendo como resultado de este análisis la identificación de los pasos que conformaron la metodología que guió el desarrollo de este trabajo.

Se implementaron todas las tareas involucradas en la limpieza de los datos así como las de los procesos analíticos mediante tres herramientas analíticas diferentes. A partir de esta múltiple implementación se ha logrado determinar aquella herramienta idónea para desarrollar un proceso analítico de Big Data.

A partir de la realización de este trabajo se logró obtener de manera exitosa información no estructurada proveniente de la red social Twitter, se manipuló dicha información para fuera almacenada de manera correcta y posteriormente fue utilizada para llevar a cabo los procesos analíticos de interés.

Como consecuencia de la implementación de los procesos analíticos que tuvieron lugar en este trabajo se logró adquirir conocimiento, el cual fue de utilidad para dar respuesta a las problemáticas planteadas en la sección 1.6. Por lo anterior, de alguna manera, podemos decir que este trabajo cumple con las contribuciones planteadas en el capítulo inicial. El grupo comercial Amazon ahora cuenta con conocimiento y soporte para la toma de decisiones sobre su novedosa propuesta comercial *Amazon Go*, cumpliendo así de esta forma el objetivo general de este trabajo.

Con base en lo explicado anteriormente, se comprueba que la hipótesis planteada en este trabajo es cierta, pues a partir de la información que se extrajo de Twitter ha sido posible identificar aquellas problemáticas que causan insatisfacción en los clientes. Para llegar a esta información se llevó a cabo una serie de procesos con ayuda de varias herramientas, entre ellas Hadoop. Concluyendo que este ecosistema es el idóneo para la implementación de un análisis Big Data.

La realización de los procesos analíticos con diversas herramientas ha permitido ratificar a Hadoop como el mejor ambiente para el procesamiento y análisis de información se que presenta de manera masiva, en lapsos de tiempos pequeños y que además carece de una estructura explícita.

A pesar de que la implementación de este trabajo se realizó con un corpus de tamaño relativamente pequeño (en comparación con los grandes volúmenes de datos que son generados en tiempo real), se han logrado establecer los pasos de una metodología para implementar un proceso analítico Big Data. Además, se identificó que debido a las características de Hadoop la arquitectura implementada en este trabajo es escalable. Retomando los enfoques para Big Data presentados en la sección 3.4.2, contando con los recursos de hardware y software necesarios; y además retomando las herramientas analíticas que se mencionaron en la sección 3.4.2, es viable implementar la arquitectura de este trabajo para que lleve a cabo el proceso analítico realizado en tiempo real.

Como trabajo futuro es viable que a partir de la identificación de las problemáticas presentadas, se continúe con la ejecución del proceso analítico pero ahora dando seguimiento al comportamiento de dichas problemáticas, lo cual automatizaría el proceso y en consecuencia se prescindiría de la identificación manual de las problemáticas a partir de los grupos formados a través de la técnica de agrupamiento.

ANEXO

A

Ilustraciones y bloques de código del capítulo 6

En este anexo se presentan ilustraciones correspondientes a diagramas de componentes y salidas de ejecuciones, así como los bloques de código utilizados en el capítulo 6.

El anexo está organizado por secciones, las cuales corresponden a las mismas que componen al capítulo 6 de este trabajo. Esto con el próposito de tener mayor claridad en lo presentado en este anexo.

A.1 Implementación de la adquisición de datos

Las siguientes figuras ilustran la creción de una aplicación a través de la cual se realizará el consumo de servicios de la API de Twitter (figura A.1).

Posteriormente, mediante la figura A.2 se ejemplifica el proceso de creación de las llaves necesarias para poder configurar la aplicación en cuestión.

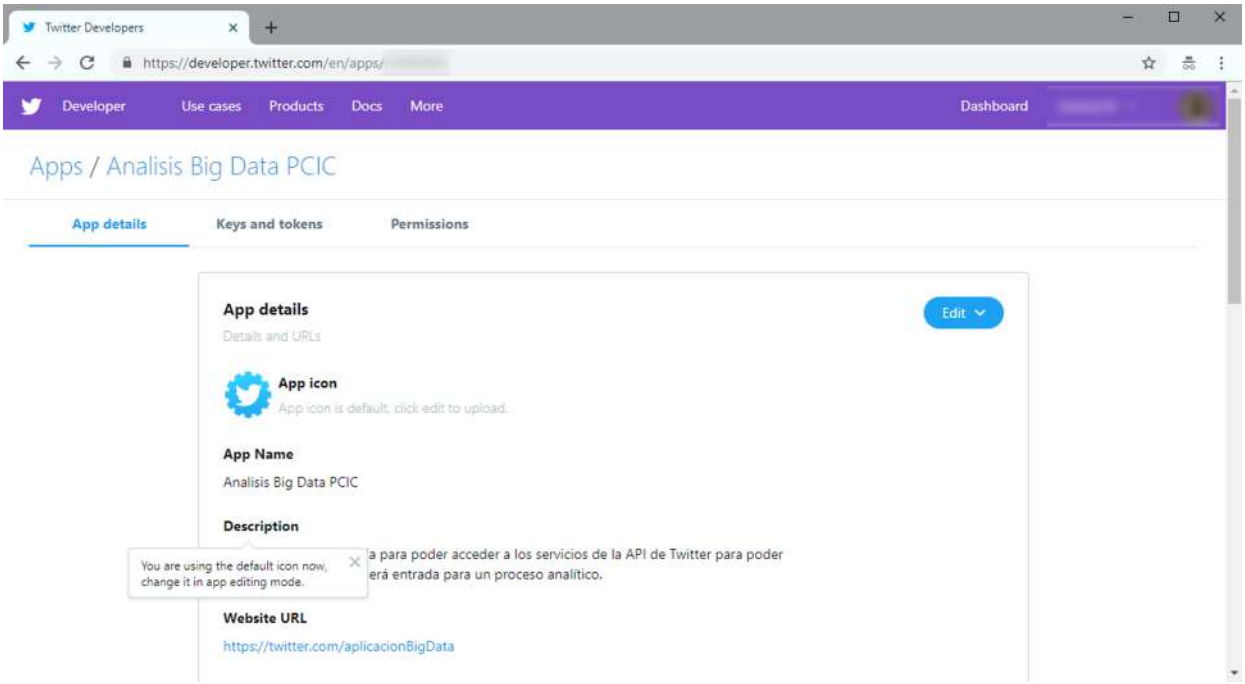


Figura A.1: Detalle de aplicación creada en Twitter.

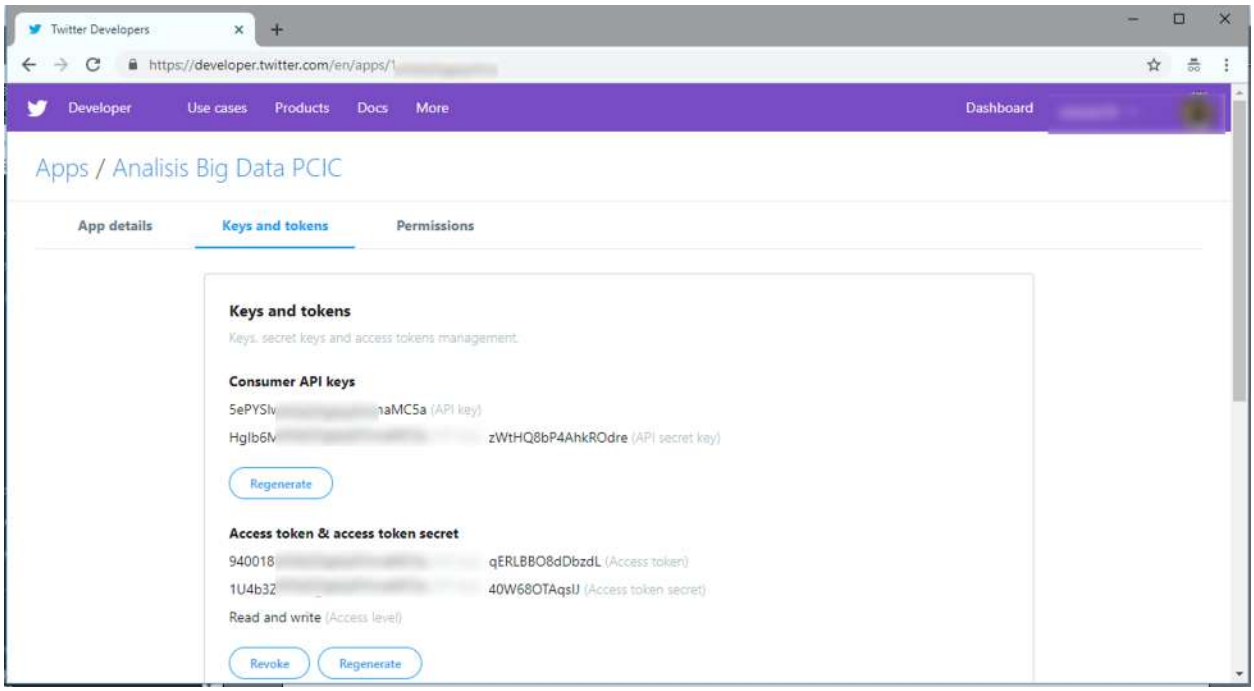


Figura A.2: *Keys and Tokens* de la aplicación creada en Twitter.

Configuración para la solicitud de peticiones a la API de Twitter.

```
consumer_key = "5ePYSIw0xxxxKqSF2maMC5a"
consumer_secret = "HgIb6MWbI0utmXFxxxxxx4woidrNP0tzWtHQ8bP4AhkROdre"
access_key = "940018738300182528-FI3yTKWxxxxxx8iqERLBBO8dDbzdL"
access_secret = "1U4b3Z9k7eogaEikxxxxxxwNon7d5HCW40W68OTAqsIJ"
```

Código A.1: Configuración para la conexión con la API de Twitter.

Solicitud de peticiones a la API de twitter.

En el bloque de código se utilizan las credenciales de la aplicación[1] creada previamente (líneas 4 y 5) para crear un objeto que será el que realice la comunicación con la API. Se establece el archivo en el cual se almacenarán los resultados de la búsqueda a través de la línea 6. En la línea 7, se declara una cabecera para dicho archivo; siendo en la línea 8 donde se hace la petición de búsqueda.

Los resultados recuperados son almacenados en el archivo indicado. Esto se realiza a través de las líneas 10 y 11, en donde mediante un ciclo *for* se recorre cada uno de los elementos dentro de la lista de resultados (contenidos en la variable *query*). En la línea 11 es donde se extraen los campos de interés para cada resultado, respetando el formato que se ha especificado en la cabecera del archivo. El script finaliza con el cierre del archivo empleado, esto se realiza mediante la línea 12.

```
from twitter import Twitter, OAuth
import sys,config
sys.path.append(".")
twitter = Twitter(auth = OAuth(config.access_key, config.access_secret,config.
    consumer_key,config.consumer_secret))
salida = open('tweets_corpus.txt','w')
salida.write('username;date;retweets;favorites;text;geo;mentions;hashtags;id;
    permalink \n')
query = twitter.search.tweets(q = "#amazongo")
for result in query["statuses"]:
    salida.write("%s;%s;%s;%s;%s;%s;%s;%s;%s;%s" %(result["user"]["
    screen_name"],result["created_at"], result["retweet_count"],result["
    favorite_count"],result["text"],result["geo"],result["entities"]["
```

[1] A través del bloque de códigoA.1

```
    user_mentions"],result["entities"]["hashtags"],result["id"],result["
    entities"]["urls"]))
salida.close()
print("END of the execution!")
```

Código A.2: Archivo de peticiones hacia la API de Twitter.

Salida de ejecución del bloque de código A.2.

En la figura A.3 se verifica que la ejecución del código terminó sin ningún problema (parte superior de la ilustración). En la parte inferior de la misma imagen se aprecia que el formato que se ha indicado en la sección 5.2.3 es consistente con la salida. En la primera línea se encuentra dicho encabezado, y partir del primer tweet (segunda línea) se respeta dicho formato. En caso de que un valor no exista aparecerá como nulo.

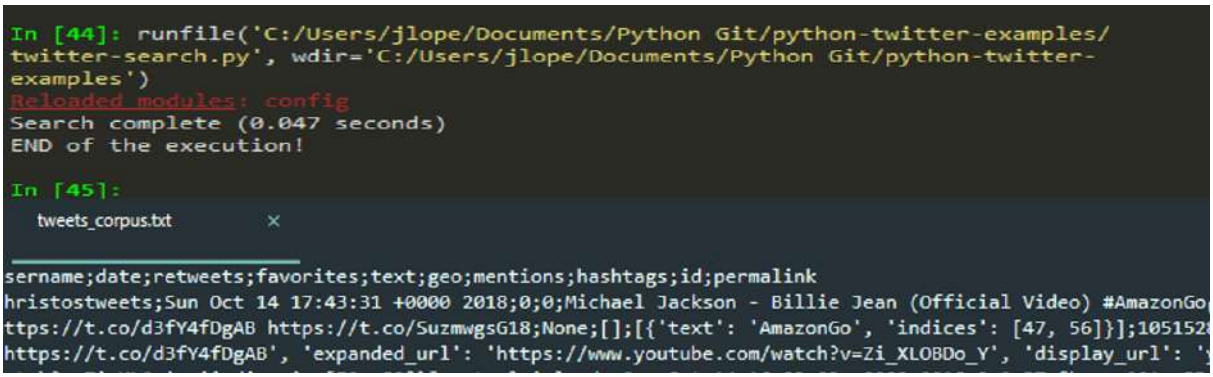


Figura A.3: Ejecución y salida del código para obtención de tweets.

A.2 Implementación de la preparación de datos

Carga del corpus en el HDFS.

Verificación de la carga correcta del corpus en el HDFS para el proceso de selección de atributos.

```
[root@sandbox Documents]# hdfs dfs -put /root/Documents/tweets_corpus
[root@sandbox Documents]# hdfs dfs -ls
Found 1 items
-rw-r--r--   1 root root    1777029 2018-10-16 00:16 tweets_corpus
[root@sandbox Documents]# 3~Btex3~Btexthdfs d
```

Figura A.4: Carga del corpus en el HDFS.

Selección de atributos en el HDFS.

La selección de atributos se realiza con el framework HIVE, el cual forma parte del ecosistema de Hadoop, en el siguiente bloque de código se encuentran las instrucciones necesarias para realizar este proceso.

Mediante la primera línea se verifica que no exista previamente las tabla *tweetText*, sobre la cual se cargará el corpus. En caso de que ya exista la tabla esta será eliminada. A través de las líneas 2 a 6 se define la creación del tabla, y se indican todos los atributos que se mencionaron el la tabla 5.1. Después de la creación de la tabla, en la línea 7 se realiza el poblado de ésta partiendo del hecho de que los datos que se han cargado previamente en el HDFS.

```
1 drop table if exists tweetsText;
2 create table tweetsText(
3   username string, date string, retweets string, favorites string,
4   text string, geo string, mentions string, hashtags string, id string,
5   permalink string  )
6 row format delimited fields terminated by '\;' ;
7 LOAD DATA LOCAL INPATH '/root/Documents/tweets_corpus' INTO TABLE tweetsText;
```

Código A.3: Definición de tabla para cargar corpus en HIVE.

Salida de ejecución del bloque de código A.3

En la figura A.5 se muestra la ejecución del script nombrado *script1.hive*, el cual contiene el código A.3 descrito anteriormente.

```
[root@sandbox thesis_implementacion]# hive -f script1.hive
18/11/15 00:05:43 WARN conf.HiveConf: HiveConf of name hive.optimize.mapjoin.mapreduce does not exist
18/11/15 00:05:43 WARN conf.HiveConf: HiveConf of name hive.heapsize does not exist
18/11/15 00:05:43 WARN conf.HiveConf: HiveConf of name hive.server2.enable.impersonation does not exist
18/11/15 00:05:43 WARN conf.HiveConf: HiveConf of name hive.auto.convert.sortmerge.join.noconditionaltask does not exist

Logging initialized using configuration in file:/etc/hive/conf/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBi
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hive/lib/hive-jdbc-0.14.0.2.2.0.0-2041-standalone.jar!/org/slf4j/
der.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
OK
Time taken: 5.86 seconds
OK
Time taken: 0.997 seconds
Loading data to table default.tweetstext
Table default.tweetstext stats: [numFiles=1, totalSize=1777029]
OK
Time taken: 1.668 seconds
[root@sandbox thesis_implementacion]#
```

Figura A.5: Creación y poblado de tabla para los datos en HIVE.

Verificación de la carga completa en HIVE.

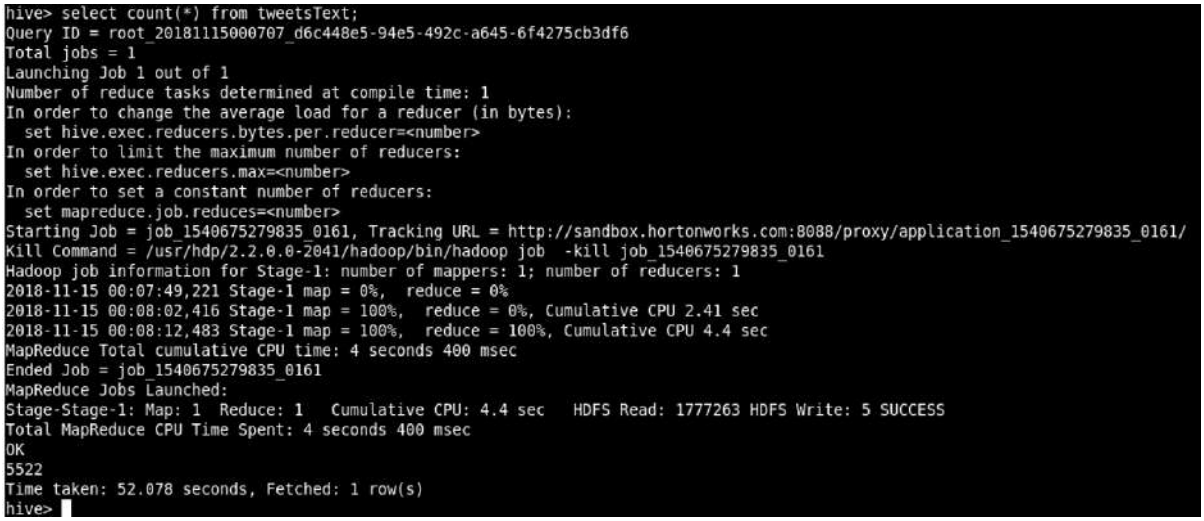


Figura A.6: Verificación de la carga correcta del corpus en HIVE.

Selección de atributos con HIVE.

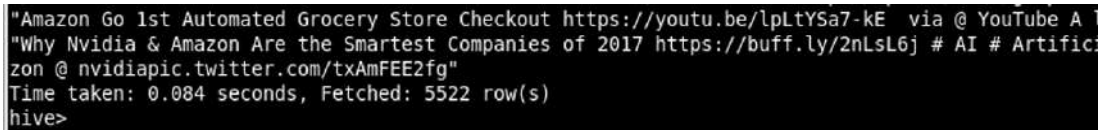


Figura A.7: Verificación de filtrado de atributos en HIVE.

Definición y poblado de tabla con propiedades transaccionales en HIVE.

A través de la línea 2, se define la tabla *tweets* la cual cuenta con las propiedades transaccionales necesarias. Después de que se ha creado dicha tabla se procede con el poblado de ésta a través de una sentencia de tipo *select*, la cual recupera los datos de la primera tabla (tweetsText), y los carga en esta nueva tabla en la línea 7 del mismo bloque de código.

Finalmente, debido a que la tabla original ya no tendrá utilidad se opta por eliminarla, lo cual se realiza a través de la última línea del código A.4.

```
1 DROP TABLE IF EXISTS tweets;
2 CREATE TABLE tweets(
3   username string, date string, retweets string, favorites string, text
    string, geo string, mentions string, hashtags string, id string,
    permalink string )
```

```
CLUSTERED BY (id)
INTO 3 BUCKETS
STORED AS ORC TBLPROPERTIES('transactional'='true');
INSERT INTO TABLE tweets SELECT * FROM tweetsText;
DROP TABLE tweetsText;
```

Código A.4: Definición y poblado de tabla con propiedades transaccionales en HIVE.

Eliminación de URLs.

A través de la consulta de tipo *update* se invoca a la función *regexp_replace*, la cual permite eliminar aquellas cadenas de caracteres correspondientes a URLs.

```
update tweets set text = regexp_replace(text,'(http(s)?://)?(www.|WWW.)
    ?[-+/[.][-a-zA-Z./$%a-zA-Z0-9]+#?_+()=%&!0-9]+','');
```

Código A.5: Eliminación de URLs con el ambiente Hadoop.

Eliminación de Hashtags.

A través de la consulta de tipo *update* se invoca a la función *regexp_replace*, la cual permite eliminar aquellas cadenas de caracteres correspondientes a hashtags.

```
update tweets set text = regexp_replace(text,'#( )?[-a-zA-Z0-9_/*#$%&()!?=@]*'
    ,'');
```

Código A.6: Eliminación de hashtags con el ambiente Hadoop.

Eliminación de usuarios.

A través de la consulta de tipo *update* se invoca a la función *regexp_replace*, la cual permite eliminar aquellas cadenas de caracteres correspondientes a usuarios.

```
update tweets set text = regexp_replace(text,'@( )?([-_a-zA-Z0-9])*','');
```

Código A.7: Eliminación de usuarios con el ambiente Hadoop.

Eliminación de caracteres no codificables.

A través de la consulta de tipo *update* se invoca a la función *regexp_replace*, la cual permite eliminar aquellas cadenas de caracteres no codificables en ASCII.

```
1 update tweets set text = regexp_replace(text,'[[^a-zA-Z0-9" "]+]','');
```

Código A.8: Eliminación de caracteres no codificables con el ambiente Hadoop.

Verificación de la eliminación de URLs, hashtags, usuarios y caracteres no codificables con Hadoop.

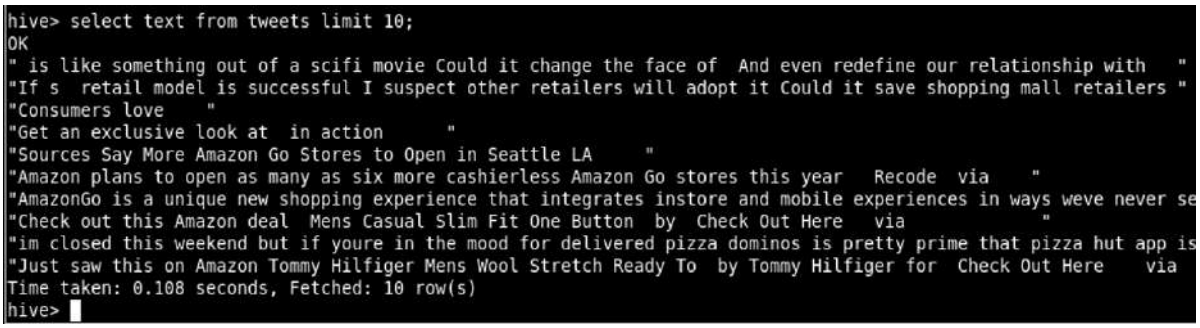


Figura A.8: Verificación de la eliminación de URLs, hashtags y usuarios con Hadoop.

Transformación del corpus en minúsculas en Hadoop.

Bloque de código para la transformación en letras minúsculas del corpus, así como para la eliminación de palabras vacías en el corpus en Hadoop. A través de la primera línea se realiza la carga de los datos desde HIVE hacia PIG. Esta línea se encarga de recuperar los tweets que se encuentran almacenados en HIVE y estos datos son almacenados en la relación *tweets*. Posteriormente, en la línea número 2 se realiza una proyección seleccionando únicamente los atributos correspondientes al identificador del tweet y el respectivo tweet. En la misma línea se renombran los atributos como *idtweet* y *texto*, respectivamente. En la última línea es donde realmente se lleva a cabo el procesamiento deseado, ya que por cada tupla que contiene al relación *tweets*, se generará una tupla cuyo contiendo estará conformado por el tweet en letras minúsculas (salida del operador *lower*) y el identificador del tweet. En la misma instrucción se utiliza el operador *tokenize* el cual toma el tweet y genera los tokens del tweet. Todas estas tuplas procesadas son almacenadas en la relación denominada *words*.

```
1 tweets = LOAD 'tweetsText' USING org.apache.hive.hcatalog.pig.HCatLoader();
2 tweet = FOREACH tweets GENERATE $1 as dateTweet, $4 AS texto, $8 AS idTweet;
3 tokens = FOREACH tweet GENERATE dateTweet,TOKENIZE(LOWER(texto)) AS token,
    idTweet;
```

Código A.9: Transformación a minúsculas y separación por palabras en el ambiente Hadoop.

Verificación de la correcta transformación en minúsculas del corpus en Hadoop.

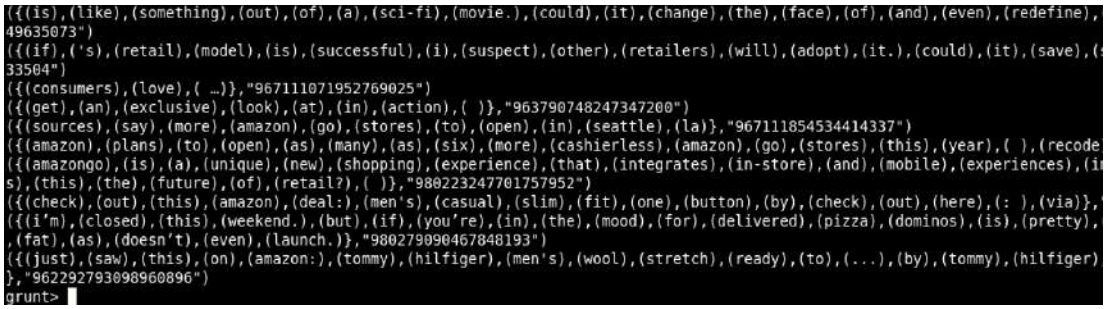


Figura A.9: Verificación de transformación a minúsculas y separación por palabras en ambiente Hadoop.

Carga de diccionario de palabras vacías en ambiente Hadoop.

```
1 stopwords = LOAD '/user/root/stopwords_dictionary' USING PigStorage ('\n');
```

Código A.10: Carga de palabras vacías en ambiente Hadoop.

Eliminación de palabras vacías en Hadoop.

A través de línea 1 del bloque de código A.11 se genera un esquema a la relación que contiene el diccionario (la relación se nombró *stopwords* en el bloque de código A.10); este esquema simplemente identifica al único atributo que tiene la relación bajo el nombre *palabra*.

Para proceder con la eliminación de las palabras vacías en el corpus se empleará la funcionalidad *JOIN* que ofrece PigLatin, este operador funciona igual que el operador *join* en el lenguaje SQL.

Del bloque de código A.9 se tiene en ambiente la relación *tokens*, la cual para cada uno de los tweets se han generado *n* tuplas (por cada token que conforma el tweet se ha generado una). Cada token se ha asociado con el identificador correspondiente del tweet para no perder el origen del token. De tal forma que la relación *token* está formado por tuplas que contienen tres atributos, siendo de la tupla de la forma *(token, id_tweet, fechaTweet)*.

Es en la línea 2 donde se declara la relación *TWEET* la cual ejecuta el operador *join*, siendo este de tipo *left*. A través de este será posible filtrar a aquellos tokens que no aparezcan dentro del diccionario. Dicho operador utiliza primero la relación *tokens*, realizando el join mediante el atributo *token*. La relación con la que se hace el join es *stopwords* mediante el atributo *palabra*. A través de este join se generan todas las combinaciones de cada token con todas las palabras del diccionario, y el join devuelve todas las tuplas de la relación izquierda, incluyendo a aquellas que no tuvieron coincidencia con la llave del join.

Filtrar a las tuplas que no se encontraron en el diccionario de palabras vacías es el siguiente proceso, para el cual se utilizará el operador *filter*. Tal como se aprecia en la línea 3 del código A.11, se eliminan las tuplas en donde el atributo *palabra* tiene un valor nulo, lo cual implicará que dicho token no se encontró dentro del diccionario. En la línea 4 del mismo bloque de código se creará un esquema para el resultado del filtrado anterior.

Para poder reagrupar los tokens de cada tweet se procede a agrupar por el identificador del tweet y lal fecha de la publicación. Esto se realiza a través de la línea 5 del mismo bloque de código. El resultado de esta agrupación se almacena bajo el nombre *GRUPOS*.

En la línea 6 cada tupla dentro de *GRUPOS* corresponde a un tweet. Aquí se extraen los tokens que pertenecen al tweet y se agrupan de acuerdo al identificador del tweet. Por último, regeneramos el tweet a partir de los token que no se encontraron en el diccionario, empleando el operador *BagToString* es posible concatenar los tokens, en dicho operador se indica que la concatenación se realice con un espacio en blanco, tal como se aprecia en la línea 7 del código A.11.

Finalmente, a través de la línea 8 el resultado de este flujo de trabajo es almacenado en el HDFS en el directorio *SALIDA_FASE_3*.

```
1 stopwords = FOREACH stopwords GENERATE $0 AS palabra;
2 TWEET = JOIN tokens BY token LEFT, stopwords BY palabra;
3 TWEET_LIMPIO = FILTER TWEET BY stopwords::palabra IS NULL;
4 TWEET_LIMPIO = FOREACH TWEET_LIMPIO GENERATE $0 as date, $1 AS token, $2 AS
    idTweet;
5 GRUPOS = GROUP TWEET_LIMPIO by (idTweet,date);
6 AUX = FOREACH GRUPOS GENERATE TWEET_LIMPIO.token, group as idTweet;
7 AUX = FOREACH AUX GENERATE  BagToString($0,' ') AS tweet, idTweet.idTweet AS
    idTweet, idTweet.date AS date;
```

```
8 STORE AUX INTO 'SALIDA_FASE_3' USING PigStorage(',','-overwrite true');
```

Código A.11: Eliminación de palabras vacías en el corpus en ambiente Hadoop.

Verificación de la preparación de los datos en Hadoop.

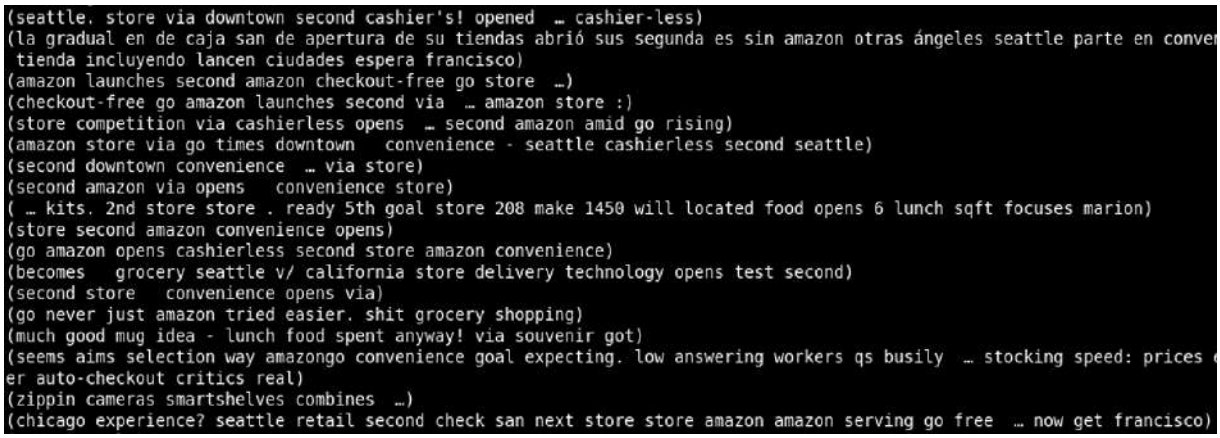


Figura A.10: Verificación del corpus preparado en ambiente Hadoop.

Selección de atributos en Python.

En la línea del bloque de código A.12 se realiza la lectura del archivo que contiene el corpus; en la dos, se define el archivo en el que se almacenará la salida de este proceso. Posteriormente, en la línea tres, se lee el contenido del corpus desde el inicio del archivo, y, a través de la línea cuatro se separa el corpus por líneas. Siendo en la línea cinco del código donde se separan los atributos de cada tweet a través del caracter delimitador *punto y coma* y estos son almacenado nuevamente en la lista *tweets*.

En la línea seis se manda un mensaje informativo al usuario indicando el número de instancias que fueron leídas en el corpus. Posteriormente, en la línea siete del código se define la variable *count*, la cual será empleada durante el proceso de limpieza. Por último antes de comenzar con el proceso de limpieza, a través de la línea ocho se elimina el primer elemento almacenado en la lista *tweets* debido a que este contiene el encabezado y no será de utilidad.

Mediante el ciclo *for* de la línea nueve tiene lugar la selección de atributos. Mientras que en la línea diez se escribe en el archivo de salida los atributos de interés. De cada tweet se utilizarán únicamente los atributos *text* y *date*, que corresponden al cuarto y primer índice respectivamente. Por ello dichos atributos son recuperados mediante las posiciones indicadas anteriormente.

Al atributo *text* se le aplica la función *strip(“”)*, la cual tiene como argumento las comillas dobles debido a la forma en la que los tweets son recuperados, ya que el mensaje se encuentra acompañado por éstas tanto al inicio como al final y la función *strip* se encarga de eliminar dichos caracteres del corpus. Por otro lado, en lo concerniente a la fecha, tal como se especificó en la tabla 5.1, este campo tiene un formato que está conformado por la fecha y hora de publicación del tweet, siendo que la hora de publicación carece de interés se rescata únicamente la fecha de publicación, esto se realiza a través de los últimos diez caracteres de la cadena correspondiente a la fecha (especificado como w[:10]).

El ciclo *for* concluye incrementando a la variable *count*, la cual funge como un contador para conocer el número de instancias que fueron procesadas al final de la ejecución de este código.

Finalmente, el bloque de código A.12 concluye con un mensaje informativo para el usuario, indicando el número de instancias que fueron procesadas (línea doce). A través de las líneas trece y catorce se cierran los archivos que fueron empleados en este bloque de código.

```
1  file = open('tweets_corpus','r',encoding="utf8")
2  output_file = open('tweets_filtrados','w',encoding="utf8")
3  file.seek(0)
4  tweets = file.read().splitlines()
5  tweets = [w.split(';') for w in tweets]
6  print("Se leyeron %s tweets" % len(tweets))
7  count=0
8  tweets.pop(0)
9  for w in tweets:
10     output_file.write(w[4].strip('"')+w[1][:10]+'\n')
11     count += 1
12 print("El filtrado ha terminado! Se limpiaron %d instancias" % count)
13 file.close()
14 output_file.close()
```

Código A.12: Selección de atributos a través de Python.

Verificación de selección de atributos en Python.

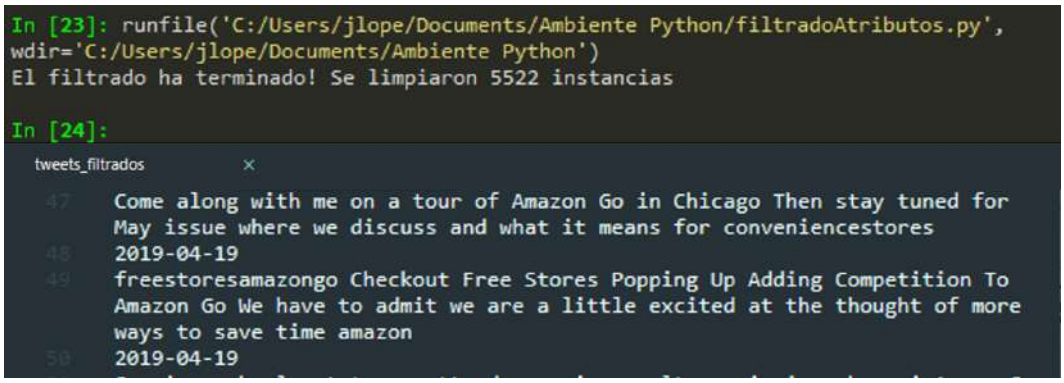


Figura A.11: Salida de la ejecución del código A.12 para filtrado de atributos con Python.

Eliminación de URLs, hashtags y usuarios en Python.

En el bloque de código A.13, a través de la primera línea se comienza con la importación de la biblioteca *re*, la cual permite emplear expresiones regulares. Posteriormente, en las líneas dos y tres, se declara y define la función *isNotBlank*; la cual recibe como parámetro una cadena. Esta función retorna un valor booleano, el cual dependerá de la cadena que se pasó como parámetro. En caso de que la cadena esté vacía, o esté formada únicamente por espacios en blancos se devolverá el valor *False*. En caso contrario, se obtendrá el valor de *True*. Esta función surge como consecuencia de que después de eliminar cadenas en el corpus, existe la posibilidad de que la cadena quede vacía, siendo esta inútil para el proceso analítico, así que lo más viable es descartar dicha cadena.

A través de las líneas tres y cuatro se declaran los archivos de entrada y salida de este flujo de procesamiento. Este último tendrá el nombre *tweets_limpios*. Siendo a través de las líneas seis y siete la lectura del corpus y separación del mismo por salto de línea. Adicionalmente, se declaran variables con fines estadísticos (líneas ocho a diez).

Después de que se han leído los tweets del corpus actual se procede a declarar y compilar las expresiones que se usarán para realizar la limpieza del corpus.

La primera expresión regular se declara en la línea once, cuya funcionalidad será eliminar las URLs que se encuentren en cada tweet. La siguiente expresión regular declarada en la línea doce del código se encarga de eliminar los hashtags y finalmente, la línea trece contiene la declaración de la expresión regular que permite eliminar las cuentas de usuario que se encuentren en los tweets.

En este bloque de código no se emplea una expresión regular adicional para la eliminación de caracteres no reconocibles, en su lugar se codifica en ASCII cada tweet ignorando aquellos caracteres no reconocidos; y posteriormente, para poder almacenar el tweet ya limpio se debe de proceder a la decodificación del mismo.

En las líneas catorce y quince se declaran las listas, la primera de ellas almacenará los tweet después de que hayan sido leídos y, la segunda, permitirá la identificación de tweets duplicados.

```
1  import re
2  def isNotBlank (myString):
3      return bool(myString and myString.strip())
4  file = open('tweets_filtrados','r',encoding="utf8")
5  output_file = open('tweets_limpios','w',encoding="utf8")
6  file.seek(0)
7  tweets = file.read().splitlines()
8  countURL=0
9  countHashtag=0
10 countTarget=0
11 patternURL = re.compile("(http(s)?://)?(www.|WWW.)?[-+/%a-zA-Z0-9]+[.][a-zA
       -Z/$#?_+()=\%&!0-9.]+")
12 patternHashtag = re.compile("#( )?[-a-zA-Z0-9_/*#$:%&()!?=@]*( )(:)*")
13 patternTarget = re.compile("@( )?([-_a-zA-Z0-9:])*")
14 listaTweets = []
15 tweets_set = []
16 for x in tweets:
17     nodo = []
18     nodo.append(x[:len(x)-10])
19     nodo.append(x[len(x)-10:])
20     listaTweets.append(nodo)
21 zero=0
22 tweet=""
23 for item in listaTweets:
24     URL = patternURL.subn("",item[0])
25     countURL += URL[1]
26     Hashtag = patternHashtag.subn("",URL[0])
27     countHashtag += Hashtag[1]
28     Target = patternTarget.subn("",Hashtag[0])
```

```
29      countTarget += Target[1]
30      tweet = tweet.encode('ascii',errors='ignore')
31      tweet = tweet.decode('ascii',errors='ignore')
32      if tweet not in tweets_set and isNotBlank(tweet):
33          tweets_set.append(tweet)
34          output_file.write(tweet+item[1]+'\n')
35          zero+=1
36  print("El proceso termino exitosamente!")
37  print("Se conservarion %d tweets" % zero)
38  print(("Los estadosticos son >> URLs = %d, Hashtags = %d y Usuarios = %d") %(
        countURL,countHashtag,countTarget))
39  file.close()
40  output_file.close()
```

Código A.13: Eliminación de URLs usuarios y hashtags a través de Python.

Es a través del ciclo *for* que se recorre cada uno de los elementos en la lista *tweets*, donde cada elemento está conformado por la concatenación del tweet y su correspondiente fecha de publicación. Por lo cual, mediante la línea dieciocho se extrae el tweet haciendo una selección desde el origen de la cadena hasta antes los diez últimos caracteres de la cadena, correspondiendo este segmento al tweet. Mientras que en la línea diecinueve se leen los últimos diez caracteres de cadena, correspondiendo estos a la fecha de publicación. Mediante la línea veinte se agrega esta pareja a la lista *listaTweets*.

Posteriormente, en las líneas veintiuno y veintidos se declaran variables que serán empleadas durante el procesamiento textual. De tal forma que en la línea veintitrés se declara un ciclo *for*, mediante el cual se van recorriendo los elementos de la lista *listaTweets*. Para cada uno de estos elementos se analiza el tweet, extrayendo en la línea veinticuatro el resultado de haber sometido la cadena original al operador *subn* con la expresión regular correspondiente a las URLs. Siendo en la línea veinticinco que se incrementa la variable *countURL* con el número de incidencias que se encontraron mediante la expresión regulada empleada.

Un proceso análogo sucede con las expresiones regulares restantes, esto se realiza mediante las líneas veintiseis a veintinueve, actualizando sus correspondientes contadores.

Mediante la línea treinta se realiza la correspondiente codificación del texto en ASCII, indicándose los errores que se puedan producir como consecuencia de caracteres no reconocibles. Inmediata-

mente se revierte este proceso en la línea treinta y uno.

Como última parte del procesamiento, mediante la línea treinta y dos se revisa que el tweet no se encuentre dentro de la lista de los tweets que ya han sido procesados (*tweets_set*) y que tampoco se encuentre vacío como consecuencia de la eliminación previa de ciertas cadenas. En caso de que las condiciones previas sean cumplidas entonces el tweet es agregado de la lista de los tweets procesados (*tweets_set*) a través de la línea treinta y tres, y también se escribe en el archivo de salida junto con su correspondiente fecha de publicación (línea treinta y cuatro); por último se incrementa en una unidad el contador *zero*, el cual lleva el conteo de tweets que han sido escritos en el archivo de salida. De esta forma se concluye la primera iteración del ciclo *for*. Este proceso se realiza para el resto de los tweets.

En las líneas treinta y seis a treinta y ocho se imprimen en consola mensajes con los estadísticos que fueron recabados durante la ejecución del proceso de limpieza. Se finaliza el código con el cierre de los archivos que fueron utilizados durante este proceso.

Verificación de la eliminación de URLs, hashtags, usuarios y caracteres no codificables a través de Python.

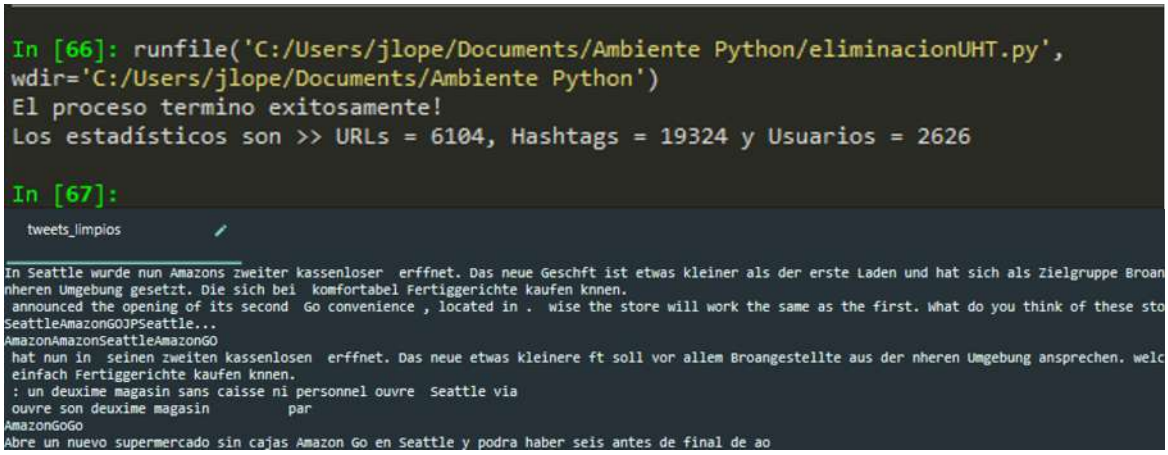


Figura A.12: Salida de la ejecución del código A.13 para eliminación de URLs, hashtags y usuarios a través de Python.

Eliminación de palabras vacías con Python.

A través de la primera línea del bloque de código A.14 se importa el diccionario con palabras vacías de NLTK, al cual se le asigna el nombre *stopwords*. Este diccionario al momento de la consulta cuenta con 179 palabras vacías.

A través de las líneas dos y tres se declaran los archivos de entrada y salida del proceso.

Posteriormente, mediante las líneas cuatro y cinco se lee todo el corpus y se separa por líneas, siendo en la línea seis donde se declara la lista *listaTweets*, la cual será utilizada para la adecuación del formato del corpus. Mediante el ciclo *for* de la línea siete se lleva a cabo la adecuación de la lectura del corpus de la misma forma en que se realizó en el bloque de código A.13. De tal forma que la lista *listaTweets* está formada por registros que tiene la estructura (tweet, fecha de publicación).

En la línea quince se realiza nuevamente la validación mediante la función *isNotBlank* definida previamente en el bloque de código A.13, la cual se encarga de revisar que, en caso de que el tweet estuviera conformado únicamente por palabras vacías, se descarte para el proceso analítico. Como parámetro de la función se reconstruye el tweet mediante la función *join*, la cual une cada uno de los miembros que conforman la lista *filtered_sentence* con un espacio en blanco.

En caso de que la validación de la función *isNotBlank* indique que la cadena no está vacía se procede a almacenar el resultado en el correspondiente archivo de salida (línea dieciséis del código).

El código finaliza con un mensaje en consola indicando que el proceso ha terminado correctamente; y posteriormente, se procede al cierre de los archivos de entrada y salida que le fueron indicados (mediante líneas dieciocho y diecinueve del código).

```
1  from nltk.corpus import stopwords
2  file = open('tweets_limpios','r')
3  output_file = open('corpus','w')
4  file.seek(0)
5  tweets = file.read().splitlines()
6  listaTweets = []
7  for x in tweets:
8      nodo = []
9      nodo.append(x[:len(x)-10])
10     nodo.append(x[len(x)-10:])
11     listaTweets.append(nodo)
12 for nodo in listaTweets:
13     filtered_words = word_tokenize(nodo[0].lower())
14     filtered_sentence = [word for word in filtered_words if word not in
       stopwords.words('english')]
```

```
    if isNotBlank(" ".join(filtered_sentence)):
        output_file.write(" ".join(filtered_sentence)+" "+nodo[1]+"\n")
print("El proceso termino exitosamente!")
file.close()
output_file.close()
```

Código A.14: Script para transformación de tweets y eliminación de palabras vacías en Python.

Verificación de la correcta transformación en letras minúsculas y eliminación de palabras vacías del corpus con Python.

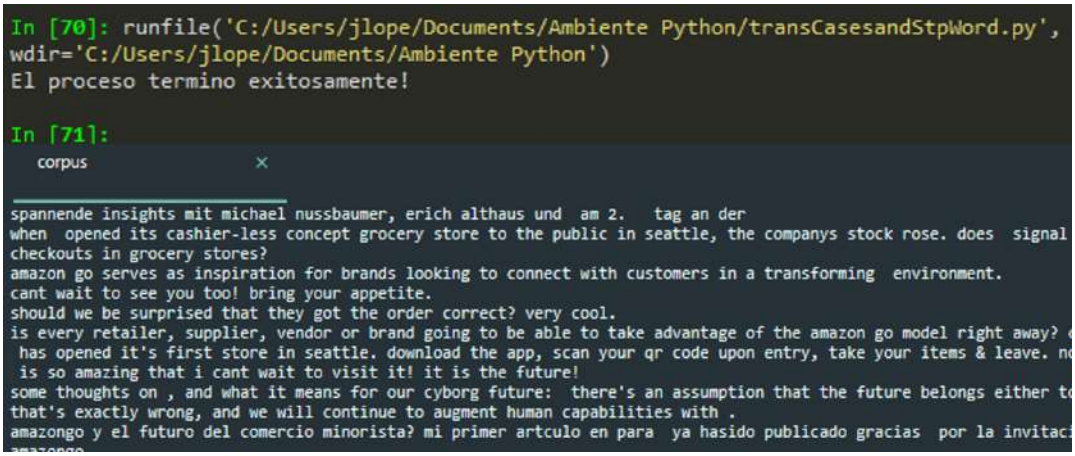
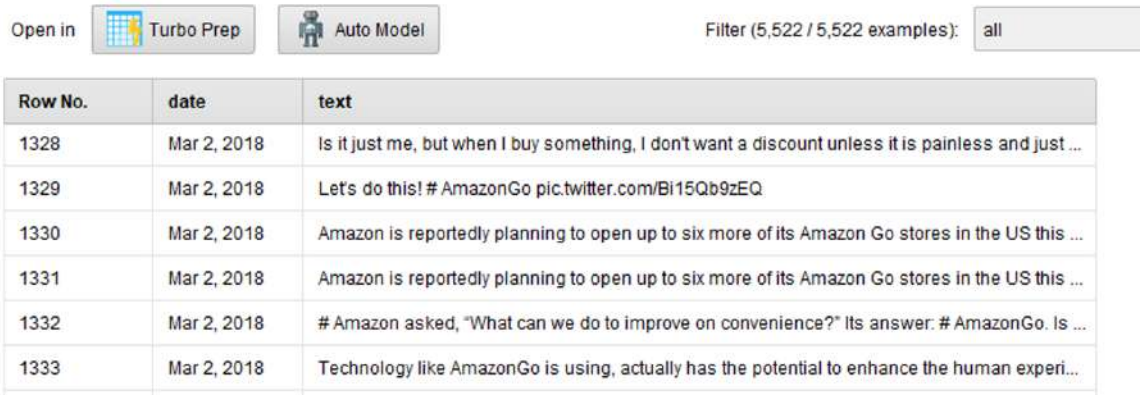


Figura A.13: Salida de la ejecución del código A.14 para transformación a minúsculas y eliminación de palabras vacías a través de Python.

Salida del filtrado de atributos en RapidMiner.



Row No.	date	text
1328	Mar 2, 2018	Is it just me, but when I buy something, I don't want a discount unless it is painless and just ...
1329	Mar 2, 2018	Let's do this! # AmazonGo pic.twitter.com/Bi15Qb9zEQ
1330	Mar 2, 2018	Amazon is reportedly planning to open up to six more of its Amazon Go stores in the US this ...
1331	Mar 2, 2018	Amazon is reportedly planning to open up to six more of its Amazon Go stores in the US this ...
1332	Mar 2, 2018	# Amazon asked, "What can we do to improve on convenience?" Its answer: # AmazonGo. Is ...
1333	Mar 2, 2018	Technology like AmazonGo is using, actually has the potential to enhance the human experi...

Figura A.14: Salida del flujo correspondiente al filtrado de atributos en RapidMiner.

Salida de la eliminación de URLs, hashtags, usuarios y caracteres no reconocibles en RapidMiner.

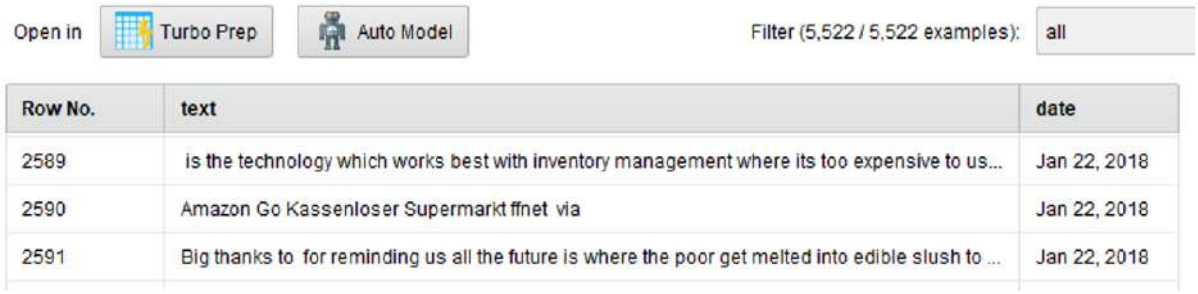


Figura A.15: Salida del flujo correspondiente a la eliminación de URLs, hashtags, usuarios y caracteres no reconocibles en RapidMiner.

Salida de la transformación en letras minúsculas y eliminación de palabras vacías con RapidMiner.

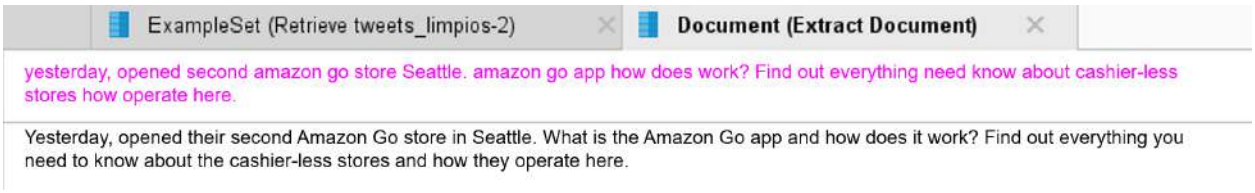


Figura A.16: Salida del flujo para la transformación y eliminación de palabras vacías con RapidMiner.

A.3 Implementación de análisis sentimental

Contenido parcial del diccionario sentimental para Hadoop.

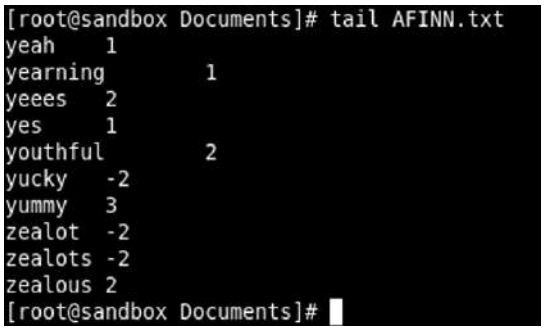


Figura A.17: Parte del contenido del diccionario sentimental.

Carga del diccionario sentimental en Hadoop.

En el siguiente bloque de código, a través de la primera línea se elimina la tabla *sentiment_analysis* en caso de que esta exista previamente. Después, en las líneas dos a seis se realiza la definición de la tabla *sentiment_analysis*. Dicha definición contiene únicamente dos atributos, tal como se mencionó, el primero de ellos corresponde a la palabra mientras que el segundo al valor. Siendo en la línea siete en donde se realiza el poblado de dicha tabla a partir del repositorio local donde se localiza el diccionario.

```
1 drop table if exists sentiment_analysis;
2 create table sentimental_analysis(
3   palabra string,
4   valor int
5 )ROW FORMAT DELIMITED
6 FIELDS TERMINATED BY '\t';
7 LOAD DATA LOCAL INPATH '/root/Documents/AFINN.txt' INTO TABLE
    sentiment_analysis;
```

Código A.15: Script para la carga de diccionario sentimental en HIVE.

Verificación de la carga del diccionario sentimental en Hadoop.

```
hive> select * from sentimental_analysis limit 5;
OK
abandon -2
abandoned       -2
abandons        -2
abducted        -2
abduction       -2
Time taken: 0.202 seconds, Fetched: 5 row(s)
hive>
```

Figura A.18: Parte del contenido del diccionario sentimental cargado en HIVE.

Análisis sentimental en Hadoop.

En el bloque de código A.16, a través de la línea uno del código se cambia al directorio en donde se almacenó la salida de la ejecución del bloque de código A.11 (a través del cual se completó la preparación de los datos). Después, en la línea dos se define la relación *tweets*, la cual contendrá el corpus limpio y, en esta misma línea se define un esquema para dicha relación, la cual está conformada por el tweet, el identificador del tweet y la fecha de publicación.

En la línea tres se define la relación *diccionario*, la cual carga en PIG el diccionario sentimental que fue cargado previamente en HIVE a través del código A.15.

```
1  cd SALIDA_FASE_3
2  tweets = LOAD 'part-r-00000' AS (tweet:chararray, id:chararray, date:chararray
       );
3  diccionario = LOAD 'sentimental_analysis' USING org.apache.hive.hcatalog.pig.
       HCatLoader();
4  tokens = FOREACH tweets GENERATE FLATTEN(TOKENIZE(tweet)) AS token, $1 AS id,
       $2 as date;
5  my_join = JOIN tokens BY token, diccionario BY palabra;
6  tweet_evaluado = FOREACH my_join GENERATE $0 AS token, $1 AS id, $2 as date, (
       $3 IS NULL ? 0 : $4) AS valor;
7  tweet_agrupado = GROUP tweet_evaluado by (id,date);
8  tweet_agrupado = FOREACH tweet_agrupado GENERATE group.id AS id, group.date AS
        date, tweet_evaluado.token AS tweet, tweet_evaluado.valor AS valor;
9  AUX = FOREACH tweet_agrupado GENERATE SUM(valor) as valor, id, date,
       BagToString(tweet,' ') as tweet;
10 cd ..
11 STORE AUX INTO 'SALIDA_FASE_4' USING PigStorage(',','-overwrite true');
12 tweets_negativos = FILTER AUX BY valor < 0;
13 A = FOREACH tweets_negativos generate id, tweet, date;
14 STORE A INTO 'TWEETS_FILTRADOS' USING org.apache.pig.piggybank.storage.
       MultiStorage('TWEETS_FILTRADOS', '0');
```

Código A.16: Script para análisis sentimental con PIG.

Teniendo en el ambiente los datos, el siguiente paso consiste en obtener las palabras que conforman cada tweet, lo cual se realiza a través del operador *tokenize*. Para poder utilizar dichas palabras como tuplas independientes es necesario pasar ese grupo de palabras por el operador *flatten*. Este proceso se realiza a través de la línea cuatro, de modo tal que la relación *tokens* estará formada por tuplas ternarias, donde el primer atributo corresponde a la palabra (token), el segundo atributo será el identificador (id) del tweet al que pertenece dicha palabra y el tercero la fecha de publicación del tweet (date).

El siguiente paso, luego de tener la relación *tokens*, consiste en implementar un *join* entre las tuplas de las relaciones *tokens* y *diccionario*; en la línea cinco del código se define la relación *my_join*, la

cual invoca un *join* cuya llave de coincidencia es el atributo *palabra* para la relación *diccionario*, mientras que para la relación *tokens* se utiliza el atributo *token*. Como nos interesan únicamente aquellas tuplas que se encuentran en el diccionario sentimental el tipo de *join* que se debe de emplear es un *natual join*.

Después, en la línea seis, de la relación resultante del *join* se obtienen los atributos que son de interés, para lo cual se define la relación *tweet_evaluado*, dicha relación está formada por la palabra, el identificador del tweet, la fecha de publicación del tweet y el valor asociado a dicha palabra con base en el diccionario. A partir de esta relación se debe de reconstruir el tweet, para lo cual en la línea siete a través del operador *group* genera grupos de palabras por cada tweet. Hasta este punto, en la relación *tweet_agrupado* se tienen tuplas de la siguiente forma «grupo_de_palabras, id_tweet, fecha_tweet, grupo_de_valores».

Para simplificar la reconstrucción de tweet, a través de la línea 8 se genera un esquema para la relación *tweet_agrupado*, conformada por el identificador del tweet, la fecha de publicación del tweet, las palabras que lo conforman (tweet) y los valores correspondientes a dichas palabras (valor).

Finalmente en la línea nueve se define la relación *AUX*, la cual genera la suma de los valores asociados a las palabras del tweet, la suma permitirá determinar la polaridad del tweet. Esta suma se define como *valor* dentro de dicha relación. El siguiente atributo que la conforma es el identificador del tweet, seguido de la fecha de publicación del tweet y por último el mismo tweet, para el cual se emplea el operador *BagToString*, dicho operador permite concatenar palabras poniendo un espacio entre cada palabra.

Antes de proceder a almacenar el resultado de este proceso analítico en el HDFS, mediante la línea diez se posiciona en directorio raíz, siendo aquí donde se alamcenará el resultado del análisis sentimental. Esta relación se almacena en el directorio *SALIDA_FASE_4* mediante el comando *store*, tal como se indica en la línea once del bloque de código A.16.

Debido a que en la ejecución del siguiente proceso analítico, de acuerdo con la sección 5.2.4, para el método de agrupamiento con lógica difusa se emplearán únicamente los tweets cuya etiqueta asignada mediante el análisis sentimental ha sido la de polaridad negativa, se procede mediante la línea doce a definir la relación *tweets_negativos*, la cual filtra a la relación *AUX* preservando aquellos tweets cuya evaluación ha sido negativa.

A partir de la selección anterior se filtran únicamente los atributos correspondientes a la fecha de publicación y el propio tweet, ya que el resto de los atributos de la relación *tweets_negativos* ya no serán de utilidad.

El código finaliza con el almacenamiento en el HDFS de los tweets negativos, esto se realiza mediante el comando *multiStorage*, presentado en la línea catorce, este comando difiere del comando utilizado en la línea once en el hecho de que el comando *multiStorage* genera un directorio para cada tweet a partir del directorio que se ha indicado (TWEETS_FILTRADOS en este caso), esto será necesario para implementar el método de agrupamiento con lógica difusa.

Verificación de la salida del análisis sentimental en Hadoop.

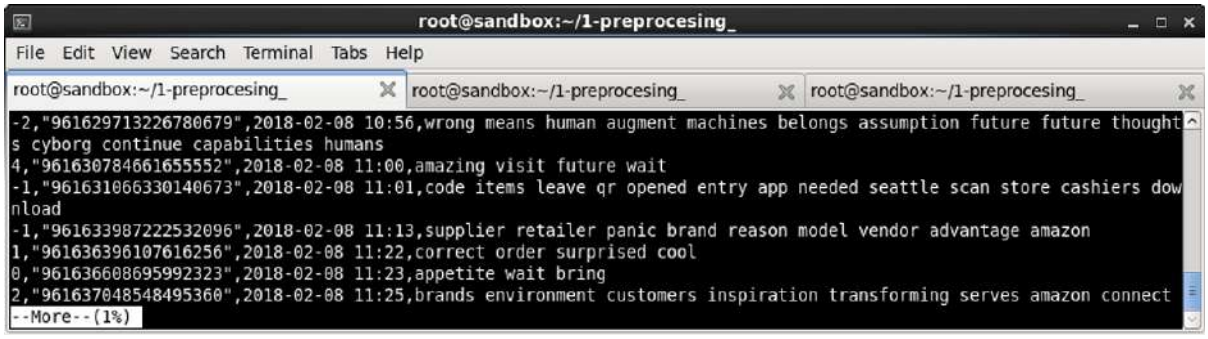


Figura A.19: Salida del análisis sentimental con PIG.

Verificación de la salida del análisis sentimental en Hadoop, tweets negativos.

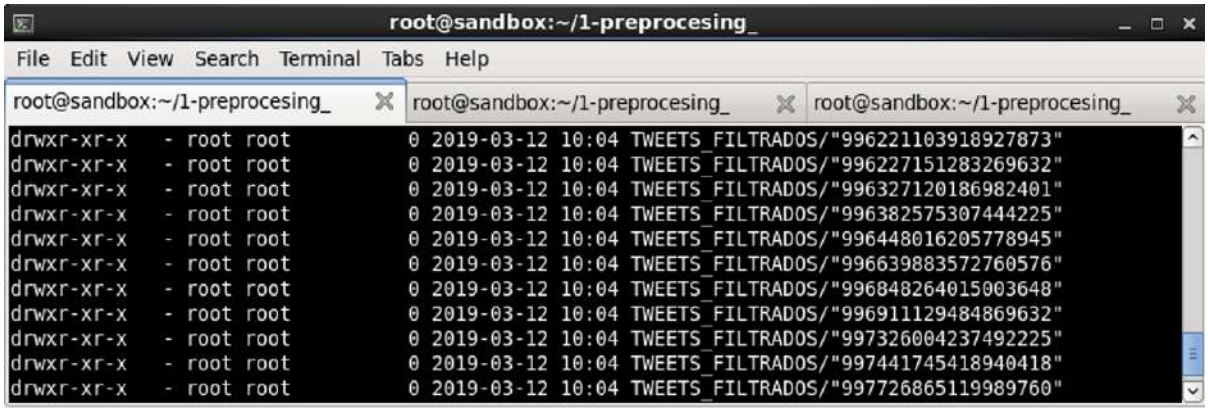


Figura A.20: Tweets etiquetados como negativos, almacenados en el HDFS.

Análisis sentimental con Python.

Para llevar a cabo el análisis sentimental con Python se empleará dicha biblioteca. En el bloque de código A.17 se presentan las instrucciones que realizarán el análisis sentimental al corpus. El primer paso es importar la biblioteca TextBlob, lo cual se realiza a través de la primer línea de código. Posteriormente, en las líneas dos a cuatro se declaran las variables que servirán como contadores globales para obtener al final del análisis un conteo global de los resultados. Después, en la línea cinco comienza la definición de la función *analize_sentiment*, la cual recibe como parámetro una cadena nombrada *tweet*. Dicha función invoca a la biblioteca TextBlob, ejecuta el método homónimo y a partir del valor retornado (almacenado en la variable *analysis*) es posible conocer la polaridad del tweet. Dicha polaridad retorna el valor de 1 cuando ésta es positiva, 0 si es neutral y -1 cuando la polaridad es negativa.

Posteriormente, en las líneas trece y catorce se declaran los archivos de entrada y salida. Es en la línea quince donde se realiza la lectura del archivo de entrada y a través de la línea dieciséis se separa el corpus por saltos de línea. La última declaración de variables a utilizar es en la línea diecisiete, la cual consistirá en una lista conformada por pares de tipo *«tweet, fecha_tweet»*.

En la línea dieciocho del mismo bloque de código se comienza a recorrer el corpus a través de un ciclo *for*, este ciclo por cada iteración crea un nodo auxiliar mediante la línea diecinueve, al cual se le agrega el tweet (línea veinte). Recordemos que de acuerdo al formato que tiene el corpus, cada línea esta formada por el tweet concatenado con su fecha de publicación, es por esto que se seleccionan únicamente los caracteres que corresponden al tweet. Posteriormente, en la línea veintiuno se obtiene la fecha de publicación y también es agregada al nodo. Este nodo que contiene al tweet y su correspondiente fecha es agregado a la lista *listaTweets*, al terminar el ciclo *for* esta lista tendrá los tweets y sus respectivas fechas, de tal forma que es viable proceder con el proceso analítico de esta sección.

Adicional a los archivos declarados en las líneas trece y catorce, se declaran tres archivos adicionales en modo escritura, esto se lleva a cabo mediante las líneas veintitrés a veinticinco. El uso de estos tres archivos facilita la manipulación del corpus por etiqueta.

El proceso analítico tiene lugar dentro del ciclo *for* declarado en la línea veintiseis, el cual recorre los elementos de la lista con el corpus. Para cada elemento de esta lista se analiza el tweet y se obtiene el valor correspondiente a la etiqueta de la polaridad contextual del tweet (línea veinti-

siete). El resultado de este análisis es almacenado a través de la línea veintiocho, se guarda en el archivo que contendrá todo el corpus etiquetado. Mediante las validaciones comprendidas entre las líneas veintinueve a treinta y siete se escribe el tweet junto con su respectiva fecha en el archivo que corresponda (existe uno para cada polaridad), además de que se actualiza el contador correspondiente, concluyendo con esta acción el ciclo *for*.

Posteriormente en la línea treinta y ocho se imprimen en consola los resultados de los estadísticos que se recabaron durante la ejecución del procedimiento; finalizando el bloque de código con el cierre de los archivos que se emplearon.

```
1  from textblob import TextBlob
2  positivo=0
3  negativo = 0
4  neutral = 0
5  def analize_sentiment(tweet):
6      analysis = TextBlob(tweet)
7      if analysis.sentiment.polarity > 0:
8          return 1
9      elif analysis.sentiment.polarity == 0:
10         return 0
11     else:
12         return -1
13 file = open('corpus','r')
14 output = open('corpus_etiquetado.csv','w')
15 file.seek(0)
16 tweets = file.read().splitlines()
17 listaTweets = []
18 for item in tweets:
19     nodo = []
20     nodo.append(item[:len(item)-11])
21     nodo.append(item[len(item)-10:])
22     listaTweets.append(nodo)
23 tweets_negativos = open('tweets_negativos.txt','w')
24 tweets_positivos = open('tweets_positivos.txt','w')
25 tweets_neutrales = open('tweets_neutrales.txt','w')
26 for item in listaTweets:
27     aux = analize_sentiment(item[0])
28     output.write("%s, %s, %s\n" %(aux,item[0],item[1]))
29     if aux == 1:
```

```
30          positivo+=1
31          tweets_positivos.write("%s\n" %(item[0]+" "+item[1]))
32      elif aux == 0:
33          neutral+=1
34          tweets_neutrales.write("%s\n" %(item[0]+" "+item[1]))
35      else:
36          negativo+=1
37          tweets_negativos.write("%s\n" %(item[0]+" "+item[1]))
38  print("Las estadisticas son: \n Positivos >> %d \n Negativos >> %d \n Neutros
        >> %d \n Total >> %d" %(positivo, negativo, neutral, (positivo+negativo+
        neutral)))
39  file.close()
40  output.close()
```

Código A.17: Script para análisis sentimental con Python.

Salida de ejecución del análisis sentimental implementado con Ppython.

```
In [27]: runfile('C:/Users/jlope/jupiter/Source/Ambiente Python/SentimentAnalysis.py', wdir='C:/Users/
jlope/jupiter/Source/Ambiente Python')
TWEET >> amazongo  , SENTIMIENTO >> NEUTRAL
TWEET >> no lines, no checkouts: will flipkart go the amazongo and tencent way? , SENTIMIENTO >> NEUTRAL
TWEET >> cette sensation trange de partir sans payer :)  amazing experience !!  , SENTIMIENTO >> POSITIVO
TWEET >> nyyoutuberyoutuberamazongo , SENTIMIENTO >> NEUTRAL
TWEET >>  amazongo, SENTIMIENTO >> NEUTRAL
Las estadisticas son :
Positivos >> 1743
Negativos >> 325
Neutros >> 3434
Total >> 5502

In [28]:
```

Figura A.21: Salida de ejecución del análisis sentimental implementado con Python.

Verificación del corpus etiquetado después de análisis sentimental con Python.

-1	excited visit amazon go even download	2018-03-31
1	amazongo unique new shopping experi	2018-03-31
0	amazon go lines checkout grab go amazo	2018-03-30
0	exchange students japan	2018-03-30
0	amazongo amazon	2018-03-30
-1	wholefoods business model never work	2018-03-30
0	buying lunch shop future amazongo scar	2018-03-30
1	future excited go	2018-03-30

Figura A.22: Verificación del corpus etiquetado después de análisis sentimental con Python.

Salida de la ejecución del análisis sentimental con RapidMiner.

text *Category*	date *Category*	polarity(text) *Category*	confidence(text) *Number*	agreement(text) *Category*	subjectivity(text) *Category*	irony(text) *Category*
Learning from the first Amazo...	2018-08-28 ...	P	100	AGREEMENT	OBJECTIVE	NONIRONIC
Wow that was easy	2018-08-28 ...	P	100	AGREEMENT	SUBJECTIVE	NONIRONIC
Have you heard of s GO store ...	2018-08-28 ...	P	100	AGREEMENT	SUBJECTIVE	NONIRONIC
Amazon Go Expands With Op...	2018-08-28 ...	NONE	100	AGREEMENT	OBJECTIVE	NONIRONIC
Welcome to the neighborhood...	2018-08-28 ...	NONE	100	AGREEMENT	OBJECTIVE	NONIRONIC

Figura A.23: Salida de la ejecución del análisis sentimental con RapidMiner.

Ejecución del método “El codo"

Mediante el bloque de código A.18 se implementará el proceso en cuestión. A través de las primeras cinco líneas se realiza la importación de las bibliotecas que serán necesarias para el funcionamiento del código. A partir de la línea seis se define la función *clustes_text(texts, clusters)*, el objetivo de esta función es ejecutar el algoritmo de agrupamiento con lógica difusa y obtener el cálculo de la inercia para dicha ejecución.

Por lo anterior,la función *clustes_text* recibe dos parámetros: el primero de ellos consiste en el texto que será procesado, y como segundo parámetro el número de grupos con el se ejecutará el algoritmo de agrupamiento en cuestión.

Dentro de esta función lo primero que se realiza es definir a la variable *vectorizer*, a través de la cual se generará un vector de características del texto que se está procesando. Dicha transformación tiene lugar mediante la línea ocho, en la cual se obtiene el vector de características.

Posteriormente, la definición del método de agrupamiento a emplear se realiza a través de la línea nueve, especificando los parámetros k (correspondiente al número de grupos), m (coeficiente de difusión) y *max_iter* (criterio de terminación del algoritmo); y es en la línea diez en donde se ejecuta dicho algoritmo, pasando como parámetro el grupo de vectores que se desean agrupar. La última parte de esta función se encarga de retornar el valor de la inercia que se obtuvo después de haber finalizado esta ejecución del algoritmo.

A partir de la línea doce se comienza a adecuar el ambiente, eliminando ventanas que pudieran existir previamente. Después en las líneas trece y catorce se lee y separa por salto de línea el corpus, mientras que en la línea quince se declara la lista *distorsions*, en la cual se almacenarán los diferentes valores de la inercia que se vayan obteniendo. En la línea dieciséis, a través de un

ciclo *for* se ejecuta el algoritmo de agrupamiento con lógica difusa con diferente número de grupos en cada ejecución. Estos valores están comprendidos dentro del intervalo de 2 a 10, y el valor de inercia obtenido para cada ejecución es almacenado dentro de la variable *distorsions*.

El bloque de código finaliza con la representación gráfica de los resultados obtenidos, mostrando la variación de la inercia con respecto a la variación del número de grupos con los que se ejecutó el algoritmo de agrupamiento en cuestión.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn_extensions.fuzzy_kmeans import FuzzyKMeans
import matplotlib.pyplot as plt
import string, collections
from nltk.corpus import stopwords
def cluster_texts(texts, clusters):
  vectorizer = TfidfVectorizer()
  tfidf_model = vectorizer.fit_transform(texts)
  km_model = FuzzyKMeans(k = clusters, m = 2, max_iter = 2000)
  km_model.fit(tfidf_model)
  return km_model.inertia_
plt.close("all")
file = open('tweets_negativos.txt','r')
tweets = file.read().split('\n')
distorsions = []
for k in range(2, 10):
    distorsions.append(cluster_texts(tweets, k))
fig = plt.figure(figsize=(15, 5))
plt.plot(range(2, 10), distorsions,'bo-', markersize=8, lw=2)
plt.grid(True)
plt.title('Curva el codo')
plt.xlabel('Numero de grupos')
plt.ylabel('Inercia')
file.close()
```

Código A.18: Script para ejecución del método *El Codo*.

Definición de variables de ambiente para Hadoop.

```
export HADOOP_CLASSPATH="/usr/lib/hadoop/*:/usr/lib/hadoop/client-0.20/*:
    $HADOOP_CLASSPATH"
```

Código A.19: Definición de entorno para Mahout.

Ejecución del comando *seqdirectory* para la creación de archivos de secuencia.

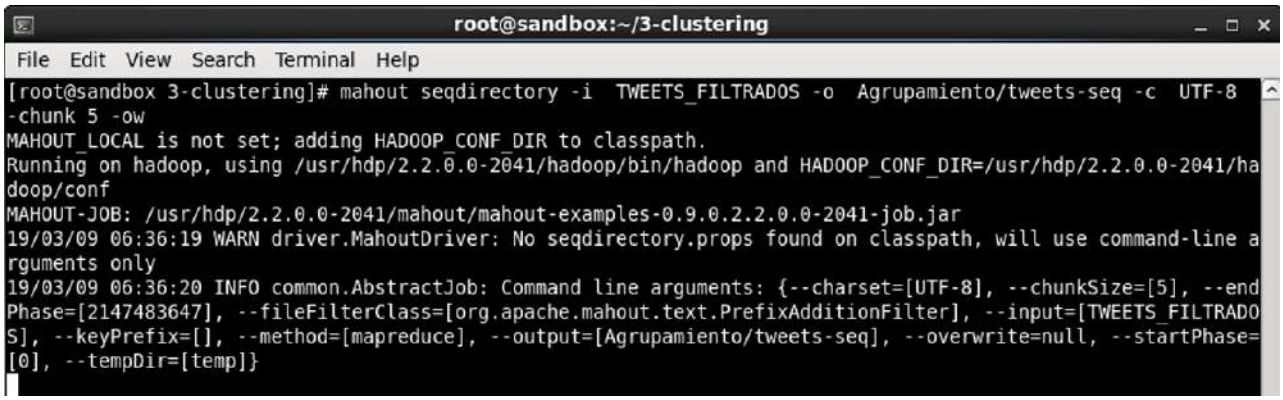


Figura A.24: Ejecución del comando *seqdirectory* para la creación de archivos de secuencia.

Verificación de la creación de archivos de secuencia en el HDFS.

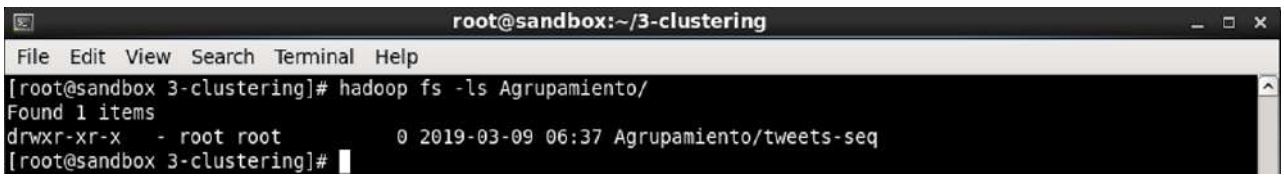


Figura A.25: Creación de archivos de secuencia en el HDFS.

Creación de vectores de características en el HDFS.

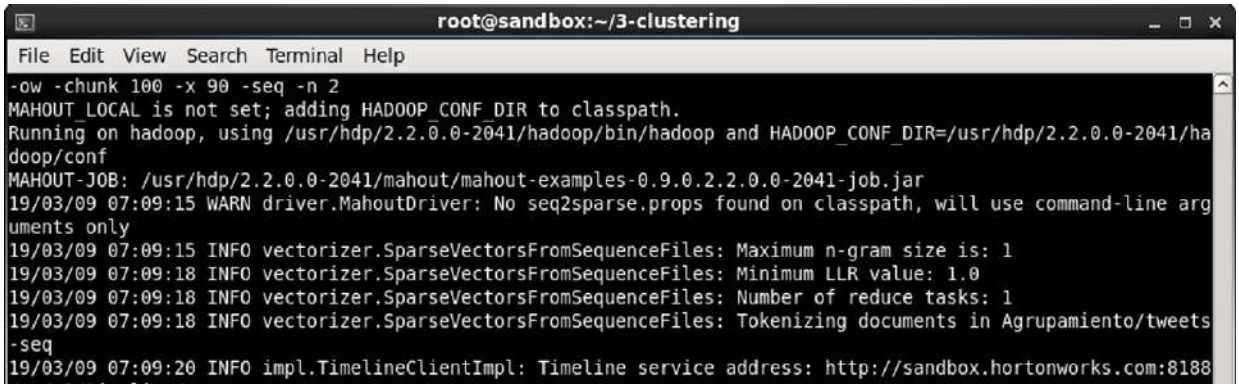


Figura A.26: Creación de vectores de características en el HDFS.

Verificación de la creación de vectores de características en el HDFS.

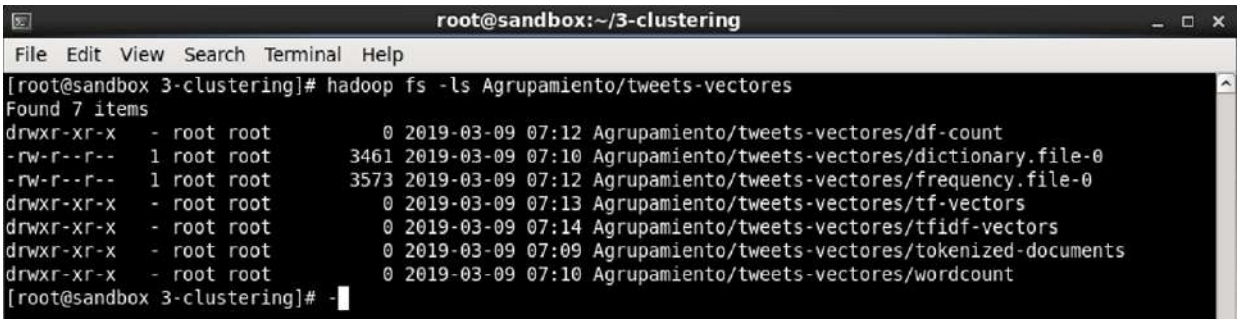


Figura A.27: Verificación de creación de archivos de vectores en el HDFS.

Verificación de la creación de archivos de centros con Canopy en el HDFS.

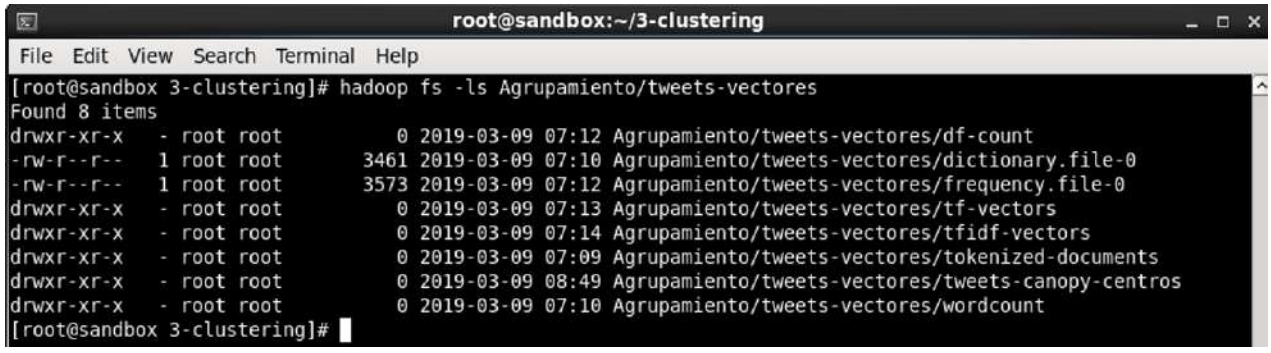


Figura A.28: Verificación de creación de archivos de centros con Canopy en el HDFS.

Ejecucíón del algoritmo de agrupamiento K-Means con lógica difusa con Mahout.

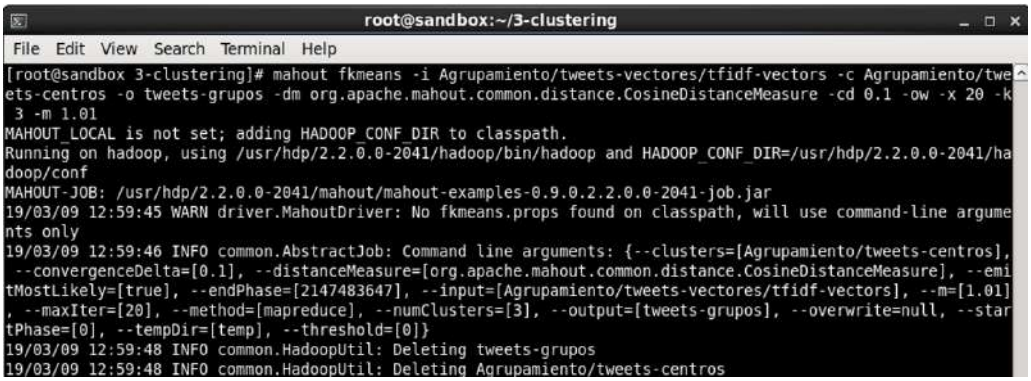


Figura A.29: Ejecucíón del algoritmo de agrupamiento K-Means con lógica difusa con Mahout.

Ejecución del algoritmo de agrupamiento K-Means con lógica difusa con Python.

Para comenzar la ejecución del algoritmo de agrupamiento se comenzará con la importación de la biblioteca mencionada previamente, tal como se aprecia en el bloque de código A.20, en la primera línea se carga en el ambiente el algoritmo a implementar, después en la segunda línea se importa una herramienta que permitirá obtener las características de los tweets.

Para la ejecución del algoritmo de agruapamiento es necesario establecer una representación numérica, ya que de este modo será posible realizar el cálculo de las distancias de cada vector (tupla) hacia los centros. Dicha representación numérica será obtenida a través de la generación de un vector de frecuencias TF-IDF (presentado en la sección 3.2.2.1). Por lo anterior, en la línea dos del mismo bloque de código se importa la herramienta *TfidfVectorizer*, la cual permitirá la generación del vector de características del corpus.

Posteriormente, en las líneas cuatro a seis del bloque de código A.20 se declaran los archivos de entrada y salida. En la línea siete donde se declara la variable *vectorizer*, la cual es necesaria ya que a través de este objeto se implementará la vectorización del corpus.

A través de la línea ocho se realiza la generación del vector de características del corpus, esta transformación es almacenada en la variable *tfidf_model*.

La declaración del modelo a emplear tiene lugar en la línea nueve, donde se invoca al modelo *FuzzyKMeans*, y ahí mismo se especifican como parámetros iniciales que $k = 3$ y $m = 2$. Donde k corresponde al número de grupos con los que trabajará el algoritmo, mientras que m corresponde

al parámetro difusión. Entre más cercano al valor de 1 este parámetro, la ejecución del algoritmo se asemejará más a *K-Means*, mientras que a un mayor valor, la difusión será mayor, convergiendo a un único grupo global.

Después de haber establecido los parámetros para la ejecución del algoritmo, en la línea diez se ejecuta la función *fit*, la cual pertenece al modelo definido en la biblioteca de Sklearn. A esta función se le pasa como parámetro la matriz de vectores de características generada previamente, se indica la opción *toarray()* debido a que esta función adecúa la presentación de los vectores de tal forma que puedan ser procesados por el algoritmo.

Para recuperar los resultados, en la línea once se declara la variable *clustering*, la cual es una colección que permitirá iterar cada uno de los grupos obtenidos, y mediante el ciclo *for* de la línea doce se recupera cada uno de los grupos con sus correspondientes vectores que le pertenecen.

Posteriormente, para realizar el mapeo de los índices contenidos en cada grupo hacia los tweets se emplea el ciclo *for* de la línea catorce, en donde cada iteración (una por cada grupo) contiene un grupo de índices, los cuales corresponden a los tweets del corpus. Es a través de un ciclo anidado *for* que se recorren los índices que pertenecen al grupo actual, y se imprimen en el archivo de salida los tweets que corresponden a los índices del grupo de la iteración actual. Mediante la línea dieciocho se imprime entre cada grupo unos saltos de línea que serán de utilidad para la legibilidad de los resultados.

El bloque de código concluye con el cierre de los ficheros de trabajo que fueron establecidos al inicio del mismo.

```
1  from sklearn_extensions.fuzzy_kmeans import FuzzyKMeans
2  from sklearn.feature_extraction.text import TfidfVectorizer
3  import string, collections
4  file = open('tweets_negativos.txt','r')
5  tweets = file.read().split('\n')
6  output = open('clustered_tweet.txt','w')
7  vectorizer = TfidfVectorizer(stop_words='english')
8  tfidf_model = vectorizer.fit_transform(tweets)
9  km_model = FuzzyKMeans(k = 3, m = 2, max_iter = 2000)
10 km_model.fit(tfidf_model.toarray())
11 clustering = collections.defaultdict(list)
12 for idx, label in enumerate(km_model.labels_):
13   clustering[label].append(idx)
```

```
for w in clustering:
  output.write("Cluster %d: \n" %(w+1))
  for x in clustering[w]:
    output.write(" %s \n" %tweets[x])
  output.write("\n\n\n")
output.close()
file.close()
```

Código A.20: Script para algoritmo de agrupamiento K-Means con lógica difusa con Python.

Salida de la ejecución del algoritmo de agrupamiento K-Means con lógica difusa con Python.

```
In [40]: runfile('C:/Users/jlope/jupiter/Source/Ambiente Python/ejemplo2-FuzzyKMeans.py',
wdir='C:/Users/jlope/jupiter/Source/Ambiente Python')
Cluster 3:
  for    it is among the  for the
wow that was easy
Cluster 1:
   is the first  rival to open an automated checkout store the bay area  has launched a small
public demo in san francisco that will grow into a fullsized  convenience store in the coming
months
yes so is amazongo lots of fixed costs associated with development but at scale the variable costs
are manageable especially with reductions in needed headcount and value of data subsidized by
brands in store retail is the new media landscape
Cluster 2:
second amazon go store opens in downtown seattle   i reside in seattle and i have not been to the
amazon go store yet but going soon
amazon opens its second checkoutfree go store in seattle
```

Figura A.30: Salida de la ejecución del algoritmo de agrupamiento K-Means con lógica difusa con Python.

Obtención de métrica de validación interna Davies-Bouldin.

Figura A.31: Obtención de métrica de validación interna Davies-Bouldin.

Resumen del modelo de agrupamiento K-Means con lógica difusa con RapidMiner.

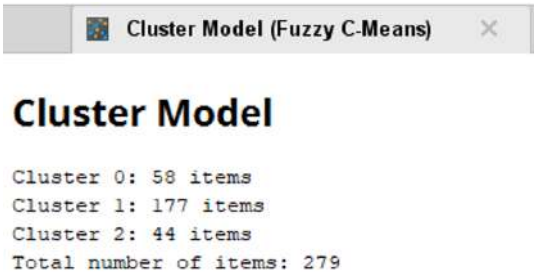


Figura A.32: Resumen del modelo de agrupamiento K-Means con lógica difusa con RapidMiner.

Salida del algoritmo de agrupamiento K-Means con lógica difusa con RapidMiner.

Row No.	text	cluster	text:abollria	text:about	text:abren	text:absolut...
235	seamless an...	cluster_2	0	0	0	0
236	is pushing to...	cluster_1	0	0.202	0	0
237	seamless an...	cluster_2	0	0	0	0
238	check out this...	cluster_0	0	0	0	0

Figura A.33: Salida del algoritmo de agrupamiento K-Means con lógica difusa con RapidMiner.

A.4 Implementación de evaluación de resultados

Script para la generación de la matriz de confusión con Python.

Teniendo en el ambiente los vectores con las polaridades (uno para los valores reales y otro para los predichos), basta con importar la biblioteca *Sklearn.Metrics*, en donde se encuentra el método *confusion_matrix*, el cual recibe como parámetros obligatorios los dos vectores mencionados previamente. Dichos vectores han sido leídos del resultado del proceso analítico en cuestión y del corpus etiquetado designado para esta sección, cuyos valores son almacenados en las variables *y_test* (para los valores reales) y *y_pred* (para las etiquetas generadas por el proceso analítico). Como parámetro opcional se pueden indicar las etiquetas, las cuales han sido definidas en la línea tres mediante la variable *labels*.

El resultado de la ejecución del método para la generación de la matriz es almacenado en la variable *cm*, esto se realiza en la línea cuatro del bloque de código en cuestión.

Este método se encarga de realizar el conteo para cada una las etiquetas, y es mediante las líneas cinco y seis que se manda a imprimir en consola la matriz que se ha calculado. La salida de la ejecución de las líneas de código descritas hasta este punto se ha acomodado de manera visual, de tal forma que se puedan entender mejor los resultados obtenidos. En la figura 6.10, se aprecian los valores obtenidos de la matriz.

```
from sklearn.metrics import confusion_matrix
import pylab as plt
labels = [1, 0, -1]
cm = confusion_matrix(y_test, y_pred, labels)
print("La matriz de confusion es:")
print(cm)
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(cm, cmap='RdBu')
plt.title('Matriz de Confusion')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Valor Predicho')
plt.ylabel('Valor Real')
plt.show()
```

Código A.21: Script para la generación de la matriz de confusión con Python.

Ejecución de técnica PCA.

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
reducedMatrix = pca.fit_transform(tfidf_model.toarray())
Matrix = reducedMatrix.transpose()
plt.scatter(Matrix[0], Matrix[1])
plt.title('Representacion de Vectores')
plt.show()
```

Código A.22: Script para la visualización de los vectores.

Determinación del desempeño para el algoritmo de agrupamiento K-Means con lógica difusa con RapidMiner.

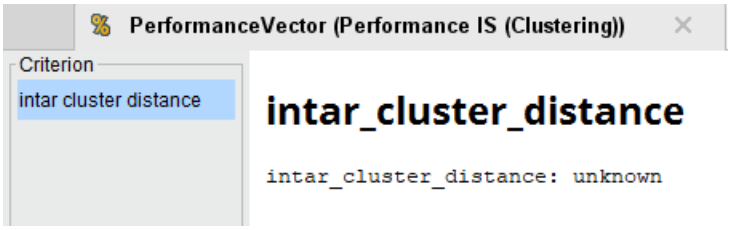


Figura A.34: Determinación del desempeño para el algoritmo de agrupamiento K-Means con lógica difusa con RapidMiner.

Distribución de los vectores en los grupos con RapidMiner.

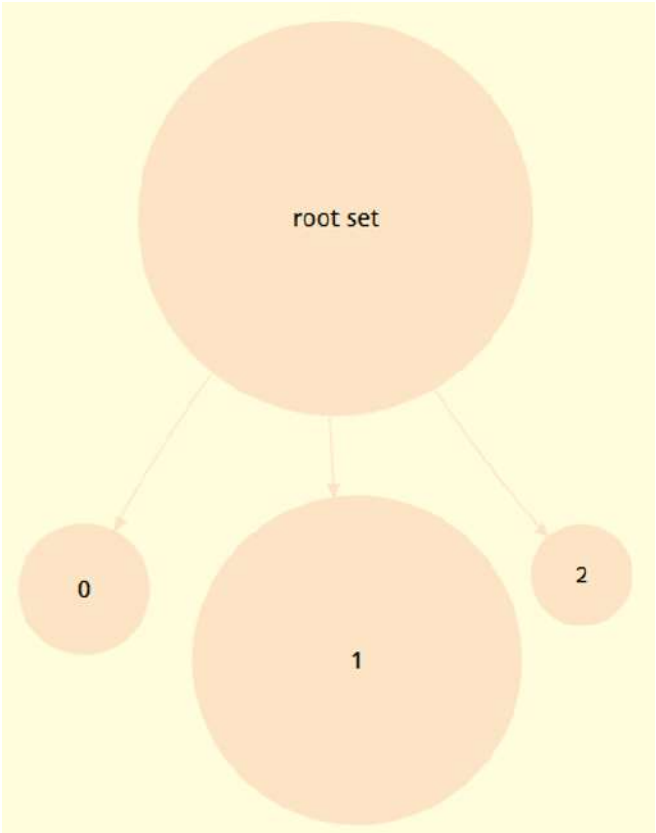


Figura A.35: Distribución de los vectores en los grupos con RapidMiner.

A.5 Implementación de visualización de resultados.

Tiempo de ejecución del tareas para preparación de datos con Hadoop.

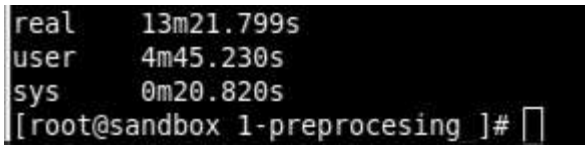


Figura A.36: Tiempo de ejecución del tareas para preparación de datos con Hadoop.

Tiempo de ejecución del análisis sentimental con Hadoop.

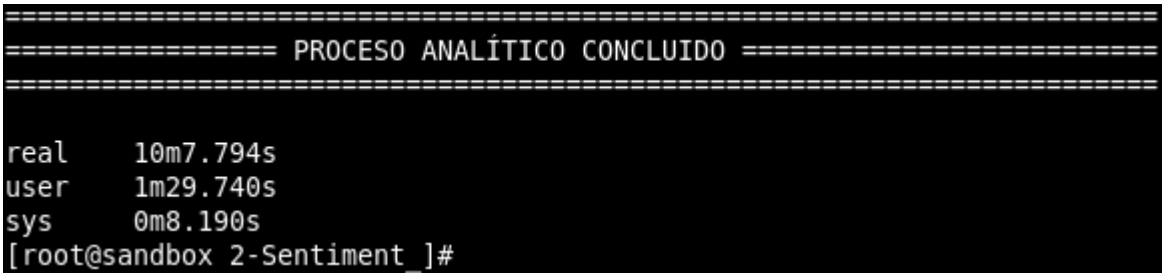


Figura A.37: Tiempo de ejecución del análisis sentimental con Hadoop.

Tiempo de ejecución del algoritmo K-Means difuso con Mahout.

Figura A.38: Tiempo de ejecución del algoritmo K-Means difuso con Mahout.

Código para la obtención de tiempos de ejecución de procesos con Python.

```
1 import os
2 from time import time
3 t0 = time()
4 print("Comenzando la limpieza de datos")
```

```
os.system("python C:/Users/jlope/jupiter/Source/2_Python/1-filtrado de
    atributos.py")
os.system("python C:/Users/jlope/jupiter/Source/2_Python/2-eliminacionUHT.py")
os.system("python C:/Users/jlope/jupiter/Source/2_Python/3-
    transCasesandStpWord.py")
p1 = time() - t0
print ("La limpieza de datos ha tomado %.2f segundos \n" % p1)
print("Comenzando con el Analisis sentimental")
os.system("python C:/Users/jlope/jupiter/Source/2_Python/4-SentimentAnalysis.
    py")
p2 = time() - (t0+p1)
print ("La ejecucion del analisis sentimental ha tomado %.2f segundos \n" % p2
    )
print("Comenzando con la ejecucion del algoritmo de agrupamiento K-Means con
    logica difusa")
os.system("python C:/Users/jlope/jupiter/Source/2_Python/5-Agrupamiento.py")
t3 = time() - (t0+p1+p2)
print ("La ejecucion del algoritmo de agrupamiento K-Means con logica difusa
    ha tomado %.2f segundos \n" % t3)
print ("La ejecucion total del flujo de trabajo tomo %d minutos %.2f segundos"
     %( (time()-t0)//60 ,(time()-t0) % 60 ))
```

Código A.23: Script para la obtención de tiempos de ejecución de procesos con Python.

Salida de en consola de la ejecución del bloque de código para obtención de tiempos de ejecución con python.

```
In [6]: runfile('C:/Users/jlope/jupiter/Source/2_Python/Tiempos.py', wdir='C:/Users/jlope/
jupiter/Source/2_Python')
Comenzando la limpieza de datos
La limpieza de datos ha tomado 29.18 segundos

Comenzando con el Análisis sentimental
La ejecución del análisis sentimental ha tomado 3.50 segundos

Comenzando con la ejecución del algoritmo de agrupamiento con lógica difusa
La ejecución del algoritmo de agrupamiento con lógica difusa ha tomado 2.40 segundos

La ejecución total del flujo de trabajo tomó 0 minutos 35.08 segundos

In [7]:
```

Figura A.39: Salida de en consola de la ejecución del bloque de código para obtención de tiempos de ejecución con python.

Bibliografía

[1] Thomas M. Conolly. *Sistemas de Bases de Datos: un enfoque práctico para diseño, implementación y gestión*. Pearson Education, 2005. 9

[2] A. Silberschatz, H.F. Korth, S. Sudarshan, and F.S. Pérez. *Fundamentos de bases de datos*. McGraw-Hill, 2006. 10

[3] W. Inmon. *Building the Data Warehouse*. Wiley Computer Publishing, 2002. 10, 12

[4] M. Kimball, R. y Ross. *The Data Warehouse Toolkit*. Wiley Computer Publishing, 2002. 11

[5] J. Han and M. Kamber. *Data Mining*. Morgan Kaufmann Publishers, 2 edition, 2006. 12

[6] Debra A. Lelewer and Daniel S. Hirschberg. Data compression. *ACM Computing Surveys (CSUR)*, 19(3):261–296, 1987. 13

[7] Víctor González Castro. *The Use of Alternative Data Models in Data Warehousing Environments*. PhD thesis, Heriot-Watt University, May 2009. 14

[8] Sin Autor. *IoT*. URL: https://www.wonderlabwoman.nl/iot/. 15

[9] A. Labrinidis and H. Jagadish. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment*, 5(12):2032 – 2034, 2012. 17, 41, 48, 50

[10] D.L. Olson. *Descriptive Data Mining*. Computational Risk Management. Springer Singapore, 2016. 18

[11] C.E. Morr and H. Ali-Hassan. *Analytics in Healthcare: A Practical Introduction*. SpringerBriefs in Health Care Management and Economics. Springer International Publishing, 2019. 18

[12] D.L. Olson and D. Wu. *Predictive Data Mining Models*. Computational Risk Management. Springer Singapore, 2016. 18

[13] G. Chandra. *Big Data Predictive and Prescriptive Analytics*. Information Science Reference, 2015. 18, 25

[14] Louis Columbus. *53% Of Companies Are Adopting Big Data Analytics*. URL: https://bit.ly/2xn06IM, 2017. 18

[15] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010. 19

[16] S Das and M Chen. Yahoo! for amazon: extracting market sentiment from stock message boards. *Manage. Sci.*, 539, 01 2001. 19

[17] Juan Ramos. Using tf-idf to determine word relevance in document queries. 01 2003. 22

[18] MarKus Hofmann and Ralf Klinkenberg. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. CRC Press, 2014. 22

[19] Trupti M. Kodinariya and Prashant R. Makwana. Review on determining number of cluster in k-means clustering. 2013. 23

[20] Doug Laney. *Gartner*. URL: https://gtnr.it/2KNFzlR, 2001. 25

[21] A. Aggarwal. *Managing Big Data Integration in the Public Sector*. Information Science Reference, 2015. 25

[22] J. Dean and S. y Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107 – 113, 2004. 26

[23] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty LEiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. *SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146, 2010. 27

[24] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. *Large Scale Distributed Deep Networks*. In *NIPS*, 2012. 27

[25] B. Hong, X. Meng, L. Chen, W. Winiwarter, and W. Song. *Database Systems for Advanced Applications*. Springer, 2013. 27

[26] K. Siddique, Z. Akhtar, Y. Kim, Y. Jeong, and E. Yoon. *Investigating Apache Hama: a bulk synchronous parallel computing framework*. Springer, 2017. 28

[27] A. Toshniwa, S. Taneja, A. Shukla, K. Ramasamy, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, M. Bhagat, S. Mittal, and D. Ryaboy. Storm @twitter. *SIGMOD*, 14(6):147 – 156, 2014. 29

[28] Tiziano De Matteis. *Introduction to Apache Storm*. URL: https://bit.ly/2rXYROu. 30

[29] Kevin Leong. Simplifying Big Data with SAP HANA and SAP Cloud Platform Big Data Services. `https://blogs.saphana.com/2017/09/26/simplifying-big-data-with-sap-hana-and-sap-cloud-platform-big-data-services/`, 2017. [Online; consultado 13-Abril-2019]. 32

[30] J. Nam, K. y Keun. Sports analytics & risk monitoring based on hana platform. *SoC Design Conference (ISOCC)*, pages 221 – 222, 2015. 33

[31] Oracle. A Comprehensive Cloud Based Platform Solution for all of your Data Integration Needs. `https://cloud.oracle.com/data-integration-platform`. [Online; consultado 13-Abril-2019]. 33

[32] Oracle. Managed Apache Kafka in the Cloud. `https://cloud.oracle.com/en_US/event-hub`. [Online; consultado 13-Abril-2019]. 33

[33] Oracle. IoT-enable Your Business Applications. `https://cloud.oracle.com/iot`. [Online; consultado 13-Abril-2019]. 33

[34] Oracle. Big Data Made Simple. `https://cloud.oracle.com/en_US/big-data-cloud`. [Online; consultado 13-Abril-2019]. 33

[35] An Oracle White Paper. *Oracle: Big Data for the Enterprise*. ORACLE, 2013. 33

[36] TERADATA. *Teradata Data Mover*. Technical report, TERADATA, 2017. Versión 16.10. 34

[37] Ian Grant. *Big data boosts Teradata growth*. URL: https://bit.ly/2xie4wi, 2011. 33

[38] Chris Kanaracus. *Teradata Buys Aster Data, Boosts 'big Data'Wares*. URL: https://bit.ly/2pe97QL, 2011. 34

[39] Y. Song. *Design and Implementation of HDFS Data Encryption Scheme using ARIA Algorithm on Hadoop*. Technical report, IEEE, 2017. 35

[40] Alex Holmes. *Hodoop in practice*. Manning, 2015. 35

[41] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2, 2014. 40, 48

[42] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passoneau. Sentiment analysis of twitter data. *Proceedings of the Workshop on Language in Social Media*, 2011. 43, 44, 48

[43] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2001. 44

[44] J. Hernández. *Introducción a la minería de datos*. Pearson Education, 1 edition, 2004. 49

[45] Hrushikesha Mohanty. *Big Data*. Springer, 1 edition, 2004. 49

[46] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining*. ELSEVIER, 4 edition, 2017. 59