PA141-06

FRAMEWORK DE PRE-PROCESAMIENTO DE DATOS EN MINERÍA DE TEXTO BASADO EN TECNOLOGÍAS DE BIG DATA

WILSON ALZATE CALDERÓN

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERÍA DE DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
2014

PA141-06

FRAMEWORK DE PRE-PROCESAMIENTO DE DATOS EN MINERÍA DE TEXTO BASADO EN TECNOLOGÍAS DE BIG DATA

Autor:
Wilson Alzate Calderón

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA
CUMPLIR UNO
DE LOS REQUISITOS PARA OPTAR AL TITULO DE
MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Director
Ing. Alexandra Pomares Quimbaya PhD
Comité de Evaluación del Trabajo de Grado
Ing. Luis Guillermo Torres MsC
Ing. Jorge Andrés Alvarado MsC
Página web del Trabajo de Grado
`http://pegasus.javeriana.edu.co/~PA141-06-PreprosMinText`

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
MAESTRÍA EN INGENIERÍA DE DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ, D.C.
Diciembre,2014

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Rubelia Calderón y José Asdrubal Alzate, mis padres:
Por traerme al mundo, por apoyarme, por creer en mi, por los sacrificios realizados y por ser siempre faro de luz en mi vida.

Alba Nidia, Darío, Yaned y Edelmira Alzate Calderón, mis hermanos:
Por ser fuente de inspiración, apoyarme y tenerme paciencia por el tiempo y energía que dejé de dedicarles.

A Yuri Milena Moreno Triana, mi novia:
Por su amor, apoyo incondicional y palabras de aliento en los momentos difíciles, así como también por su aporte de conocimientos en la realización de la validación del presente proyecto.

A la Ing. Alexandra Pomares PhD, mi tutora:
Por su guía y dedicación en la correcta definición y dirección del presente proyecto.

A Heinsohn Business Technology, la empresa en la cual trabajo:
Por la flexibiliad que me brindaron siempre para asistir a mis clases sin presentarme nunca un obstáculo para cumplir este sueño.

Índice

Índice de figuras

Índice de cuadros

ABSTRACT

Into the Health Care industry, the Electronic Medical (Health) Records (EMRs or EHRs) management can be classified as a Big Data problem because they have the three fundamental characteristics of that paradigm: Volume, Variety and Velocity. A big inconvenient is that the bigest part of the information (the most relevant) is found into the narrative text fields, having the Text-Mining as the key technology for knowledge discovering in that kind of problems. Within the tasks performed into a Text-Mining project, the more critical in terms of performance (time consuming) are the pre-processing ones, having there a big opportunity for optimizing processes using Big Data technologies in order to help speeding up the response times. In the present article is presented BigTexts: a framework that offers pre-built functions for the execution of pre-processing tasks in Text-Mining based on Big Data technologies.

RESUMEN

En la industria de la salud la gestión de historias clínicas electrónicas se puede clasificar como un problema de Big Data ya que cuenta con las tres características fundamentales: Volumen (gran cantidad de exámenes, notas, tratamientos, diagnósticos etc.), Variedad (datos estructurados y no estructurados) y Velocidad (necesidad de grandes velocidades de procesamiento al realizar ciertos análisis). Un gran inconveniente que se tiene es que la mayor parte de la información se encuentra en los campos de texto narrativo, siendo la minería de texto una tecnología clave para el descubrimiento de conocimiento. Dentro de las tareas que se realizan en los proyectos de minería de texto las más críticas en términos de consumo de tiempo son las de pre-procesamiento y de análisis, por lo que se encuentra allí una gran oportunidad de optimización de procesos utilizando tecnologías de Big Data que permitan agilizar los tiempos de respuesta. En el presente trabajo se presenta BigTexts: un framework que ofrece funcionalidades pre-construidas para la ejecución de tareas de pre-procesamiento en minería de texto basado en tecnologías de Big Data.

RESUMEN EJECUTIVO

En la actualidad, el monto de información almacenada en el mundo se estima alrededor de 1.200 exabytes (10^{18} bytes) [26], los cuales son producidos por múltiples fuentes; por ejemplo, Google procesa más de 24 petabytes (10^{15} bytes) de datos por día y Facebook obtiene más de 10 millones de fotos cada hora [26]. Teniendo en cuenta estas grandes cantidades de datos que se están generando actualmente, surge el paradigma de Big Data, que es un concepto generalmente usado para describir montos vastos de datos diversos, tanto estructurados como no estructurados, a los cuales las organizaciones pueden acceder de manera rápida para analizarlos usando herramientas innovadoras, que en conjunto ayudan a determinar con precisión oportunidades de mejora en la gestión y generación de valor [30].Un problema de Big Data es aquel que cumple con las siguientes tres características: Volumen (en términos de grandes cantidades de datos), Variedad (datos estructurados y no estructurados) y Velocidad (flujo de datos o de procesamiento de los mismos) [27].

El cuidado de la salud, en común con muchos otros dominios, está generando grandes cantidades de datos rutinarios, los cuales pueden ser minados e incluso combinados con tweets y blogs. Es un gran reto procesar, analizar y conservar dicha masa de datos. Darle sentido a esa gran cantidad de información ofrece oportunidades para el mejor tratamiento de una enfermedad, el mejor abordaje de temas de salud pública o el funcionamiento eficiente de los proveedores de servicios de salud [24]. El análisis de datos contenidos en las historias clínicas electrónicas usando técnicas computacionales es un problema de Big Data [31] debido al gran volumen de información contenida, a la variedad entre datos estructurados y no estructurados, así como también, a la velocidad con la que se requieren ciertos análisis.

Los campos narrativos de las historias clínicas electrónicas son fuentes ricas de información, siendo las técnicas de minería de texto las más indicadas para el análisis de dicho contenido. Algunas de las tareas relacionadas a la minería de texto son el pre-procesamiento de colecciones de documentos (categorización de texto, extracción de información, extracción de términos), el almacenamiento de representaciones intermedias, técnicas para analizar dichas representaciones intermedias (como análisis de distribución, clustering, análisis de tendencias y reglas de asociación), así como la visualización de los resultados [11].

Las operaciones de pre-procesamiento se centran en la identificación y extracción de características representativas para documentos en lenguaje natural y son responsables de la transformación de datos no estructurados almacenados en colecciones de documentos a un formato intermedio explícitamente estructurado [11]. Siendo tanto las tareas de pre-procesamiento como las del núcleo, las dos áreas más críticas para cualquier sistema de minería[11] por lo que se encuentra allí una gran oportunidad de optimización

de procesos utilizando tecnologías de Big Data que permitan agilizar los tiempos de respuesta de los sistemas de minería.

Ahora bien, la atención de un paciente en una institución de prestación de servicios de salud puede generar en un día más de 100 registros nuevos asociados a la información de medicamentos, procedimientos y observaciones. Hecho que si se considera a largo plazo, en un hospital de primer nivel, puede generar millones de registros no estructurados (observaciones y procedimientos en texto narrativo) que se deben procesar. Dichas cantidades de información son un activo importante para el análisis de tendencias [6], pero con las tecnologías tradicionales, los procesos de extracción de conocimiento o de clasificación pueden tardar demasiado tiempo, lo que va en contraposición de lo que el sector necesita; que la velocidad de respuesta sea lo suficientemente rápida para lograr obtener la información correcta en el momento preciso [7]. Además de ello, aunque existen en el mercado algunas tecnologías asociadas al paradigma de Big Data, como Hadoop [8], Cassandra, HBase, Hive, NoSQL [7] o Cloud Computing [9], con las cuales se puede llegar a abordar las complejidades de Volumen, Variedad y Velocidad del análisis de historias clínicas electrónicas, su orquestación y gestión puede llegar a ser un tanto complejas incluso para un profesional con dominio técnico. Por otro lado, se han realizado aproximaciones en el uso de tecnologías de Big Data para trabajar procesos core de minería de datos, tanto usando los tweets públicos [12], como noticias o patentes [13], pero no en análisis de historias clínicas electrónicas.

Teniendo en cuenta la problemática antes descrita se llega a la pregunta de ¿Cómo generar un framework que ofrezca funcionalidades pre-construidas para ejecutar tareas de pre-procesamiento en minería de texto basado en tecnologías de Big Data, en donde se logre adicionar un nivel de abstracción a la complejidad que implica tener que enlazar las diferentes tecnologías disponibles y así poder mejorar los tiempos de procesamiento en sistemas de minería de texto?, para lo cual este proyecto genera un modelo de aplicación de Big Data que soporta la fase de pre-procesamiento en los proyectos de minería de texto a través de un *framework*.

Utilizando la metodología de la ciencia basada en el diseño como marco general para el proyecto, se inicia con una revisión de técnicas, métodos y herramientas relacionadas tanto a los campos de la minería de texto como de Big Data. Se seleccionan algunas de ellas para tener como resultado una arquitectura de software para abordar el problema. Posteriormente, utilizando la metodología Personal eXtreme Programming, se construye un prototipo del *framework*. Se instala un clúster de máquinas en el cual es validado el sistema con un caso de estudio basado en análisis de historias clínicas electrónicas, específicamente en el análisis de notas de ingreso, notas de enfermería y plan de tratamiento del Hospital Universitario San Ignacio.

1. INTRODUCCIÓN

Actualmente se están generando grandes cantidades de datos en el mundo. Sensores en automóviles, trenes, aviones, turbinas de energía eólica, transformadores de energía eléctrica, entre otros, generan grandes cantidades de datos que son difíciles de analizar y procesar. El dominio de la salud no es ajeno a esta problemática, pues los exámenes médicos que se practican (tomografías, radiografías, electro cardiogramas, etc.) son fuentes ricas de datos, que para ser analizados en detalle, requieren técnicas complejas de computación y gran poder de procesamiento. En dicho dominio, más específicamente en la gestión de Historias Clínicas Electrónicas (EHR por sus siglas en inglés), también se tiene una fuente vasta de datos, ya que la atención de un paciente puede llegar a generar cientos de registros en un sólo día. Para el análisis de una sola historia clínica, bastaría con imprimirla y que un médico la analizara, pero si esto se extrapola a cientos, miles o millones de registros clínicos, los análisis de tendencias plantean un problema de grandes magnitudes, el cual se puede catalogar como un problema de Big Data, pues maneja las tres características fundamentales de dicho paradigma:

- Volumen: Grandes cantidades de información contenida en las historias clínicas electrónicas.
- Variedad: Datos estructurados y no estructurados
- Velocidad: Visto desde la perspectiva de la necesidad de contar con grandes velocidades de procesamiento.

El campo de observaciones de las historias clínicas electrónicas son una fuente bastante rica de información y para su tratamiento las técnicas más adecuadas son las de minería de texto. Una de las fases más pesadas en un proceso de minería de texto es la fase de pre-procesamiento y es allí en donde se encuentra una gran oportunidad de afinamiento mediante el uso de tecnologías asociadas al paradigma Big Data. En el presente documento se expone BigTexts, un framework para pre-procesamiento de datos en minería de texto soportado en tecnologías de Big Data. A continuación se presentan los objetivos del proyecto (Secciones 1.1 y 1.2).

1.1. Objetivo General

Generar un *framework* que ofrezca funcionalidades pre-construidas para ejecutar tareas de pre-procesamiento en minería de texto basado en tecnologías de Big Data.

1.2. Objetivos Específicos

1. Seleccionar las técnicas, métodos y herramientas asociados a Big Data que sean relevantes para realizar tareas de pre-procesamiento en minería de texto.

2. Diseñar la arquitectura de un framework que ofrezca funcionalidades pre-construidas para ejecutar tareas de pre-procesamiento en minería de texto basado en las tecnologías de Big Data seleccionadas.

3. Construir un prototipo del framework que implemente la arquitectura diseñada.

4. Validar las funcionalidades provistas por el framework a través de su utilización en un caso de estudio aplicado al análisis de historias clínicas electrónicas del Hospital Universitario San Ignacio.

A continuación se presenta la metodología usada en el proyecto (Capítulo 2), se realiza una revisión de lo que es el paradigma de Big Data (Capítulo 3), con una exploración de lo que existe actualmente en cuanto a su aplicación al dominio de la salud (Sección 3.1), las tecnologías asociadas al término (Sección 3.2), las herramientas o implementaciones relacionadas al paradigma (Sección 3.3) y finalmente se propone a partir del análisis de los trabajos relacionados una arquitectura de referencia (Sección 3.4).

Posteriormente, se realiza una revisión del campo de la Minería de texto (Capítulo 4)), ahondando en las técnicas de pre-procesamiento (Sección 4.1), las aplicaciones de Big Data en dicho dominio (Sección 4.2) y las herramientas existentes para realizar dichos análisis (Sección 4.3).

Luego de ello se presenta BigTexts (Capítulo 5): El *framework* de preprocesamiento de datos en minería de texto basado en tecnologías de Big Data, sus principales características (Sección 5.3), su arquitectura (Sección 5.4) y la forma en la cual se realizó la construcción del mismo (Sección 5.5).

Finalmente, se muestran los resultados de la validación del sistema (Capítulo 6), las conclusiones y trabajos futuros (Capítulo 7) así como también el listado de anexos del presente documento.

2. METODOLOGÍA

A continuación se presenta la metodología marco usada en el proyecto, la ciencia basada en el diseño:

2.1. Ciencia basada en el diseño

La Ciencia basada en el Diseño permite abordar problemas de investigación en informática y ciencias de la computación[18], cuyo objetivo es contribuir con soluciones novedosas a problemas que aunque son relevantes, no han sido resueltos. Esta metodología establece un proceso iterativo, de constante retroalimentación y mejoramiento que se compone de tres ciclos estrechamente relacionados:

1. Rigor: En este ciclo se desarrolla una revisión de los principios científicos, teorías y experiencias de investigaciones anteriores consignadas en bases de datos especializadas. Para el contexto del proyecto, se verificaron las teorías, técnicas, tecnologías y algoritmos relacionados con el pre-procesamiento en minería de texto. Luego se revisaron las técnicas y tecnologías de Big Data que pudiesen ayudar a mejorar los tiempos de respuesta. El objetivo de esta fase es identificar cuáles funcionalidades debe proveer el *framework* para soportar la ejecución de tareas de pre-procesamiento de minería de texto, mejorando la velocidad de procesamiento mediante tecnologías de Big Data.

2. Relevancia: Se continúa con un análisis del contexto del problema, en este caso, de las necesidades de la comunidad de minería de texto en cuanto a velocidades de pre-procesamiento, así como también de la organización y del entorno tecnológico del Hospital Universitario San Ignacio alrededor del análisis de historia clínica electrónica. Gracias a esta fase fué posible identificar los insumos que permitieron validar el *framework*. Estos insumos incluyen las fuentes de datos de las que provienen los textos y las características de los mismos.

3. Diseño central: Es la construcción y evaluación de teorías o artefactos, teniendo como materia prima tanto los resultados de las fases de rigor como los de relevancia, esto con el fin de generar una contribución a la base de conocimiento y también una aplicación que sea relevante en el contexto del problema.

A continuación se presenta un diagrama de la adaptación del paradigma de la ciencia del diseño aplicada al proyecto (Figura 1).

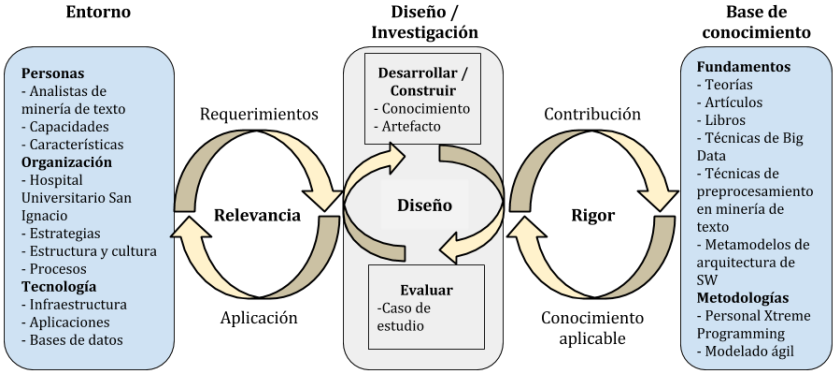


Figura 1: Ciencia basada en el diseño (Adaptada de: [18])

Las fases de la ciencia del diseño para el proyecto son:

Fase 1: Análisis de la base de conocimiento (Rigor): Inicialmente se realizó una revisión de los métodos y herramientas, basados en teorías, artículos, libros y tecnologías tanto de pre-procesamiento en minería de texto como de Big Data con el fin de identificar las funcionalidades que debía proveer el *framework*.

Fase 2: Análisis del entorno (Relevancia): Se evaluaron las necesidades de la comunidad de minería de texto en cuanto a pre-procesamiento de grandes cantidades de contenido narrativo. Se identificaron además los insumos y el entorno tecnológico del caso de estudio (Hospital Universitario San Ignacio) con el cual se realizó el *framework*.

Fase 3: Diseño, construcción y validación del *framework* (Diseño central): Esta fase se centró en el diseño de la arquitectura y la construcción de un prototipo del *framework* que ofreciera funcionalidades generales y pre-construidas para ejecutar tareas de pre-procesamiento en minería de texto basado en tecnologías de Big Data, tomando los resultados tanto de las fases de rigor como de relevancia, para luego validar el prototipo en un caso de estudio aplicado al Hospital Universitario San Ignacio.

Vale la pena indicar que ciertas actividades de las Fase 1 y 2 NO son secuenciales y se convierten en un proceso iterativo en el cual se refinan unas actividades de la Fase 1 con otras de la Fase 2.

La construcción del sistema se realizó usando la metodología Personal eXtreme Programming, ya que se identificó que en muchos de los sistemas realizados como parte de un trabajo de grado individual, se intentaba hacer uso de metodologías que no necesariamente estaban diseñadas para un equipo de desarrollo de una sola persona, fue por ello que se investigó una metodología, que fuera ágil y que se adaptara a este tipo de trabajo. A continuación se presenta un breve repaso de los componentes de dicha metodología

2.2. Personal eXtreme Programming

La metodología PXP, desarrollada en 2009 en la Sofia University, Bulgaria por Yani Dzhurov, Iva Krasteva y Sylvia Ilieva, tiene como objetivo "mejorar el rendimiento y la calidad de los ingenieros autónomos mediante la automatización de las actividades diarias de los desarrolladores y la realización de retrospectivas regulares"[10]. Algunas de las ventajas de esta metodología son la reducción del número de scripts y formas a llenar con respecto a PSP, diseñado para desarrolladores autónomos, es iterativo, permite al desarrollador ser más flexible y tener mejor respuesta a los cambios y la generación de entregas periódicas al cliente[2]. Las Fases de PXP se presentan en la figura 2:

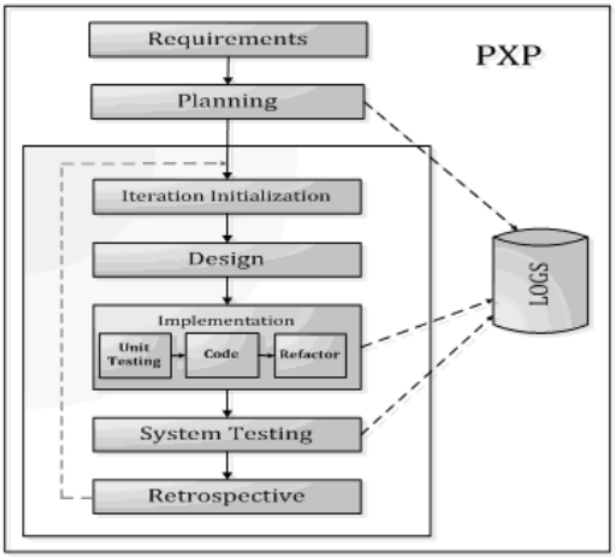


Figura 2: Fases del proceso de PXP (Tomada de:[10])

Se inicia con un documento de requerimientos y una planeación general para el proyecto, continúa con un enfoque iterativo en el cual se realiza el diseño, se utiliza la metodología Test Driven Development[5][1] en la implementación, se realizan las pruebas funcionales del sistema y al final del ciclo se realiza el análisis del proceso (Retrospectiva), que en conjunto con las bitácoras generadas sirven para mejorar en la nueva iteración o en los próximos proyectos[10].

Las prácticas de PXP se toman de dos fuentes[10]:

- PSP
    - Registro de tiempos

[1] Se realiza primero la prueba unitaria, luego se desarrolla el código necesario para pasar dicha prueba y se realiza un refactor para tener en cuenta otros escenarios

  - Medición de tamaño
  - Estándar de tipos de defectos
  - Propuesta de mejora de procesos
  - Registro de defectos
  - Revisiones de código
- XP
  - Simplicidad en el diseño
  - Pequeñas entregas
  - Refactor
  - Test Driven Development
  - Soluciones rápidas

Y cuenta con los siguientes principios o valores[10]:

- Requiere un enfoque disciplinado, y que los desarrolladores sean responsables de seguir el proceso y aplicar las prácticas de PXP.
- Los desarrolladores deben medir, almacenar y analizar su trabajo diario.
- El desarrollador debe aprender de las variaciones de su propio rendimiento y enfocarse en la mejora del proceso basado en los datos recolectados durante el proyecto.
- PXP involucra las pruebas continuas.
- La corrección de defectos debe realizarse en la etapa de desarrollo más temprana posible, donde el costo es menor.
- Los desarrolladores deben tratar de automatizar lo más que puedan de su trabajo diario.

Dentro de las métricas utilizadas en Personas eXtreme Programing encontramos[10]:

- Esfuerzo de planeación: El tiempo invertido en la fase de planeación del proceso.
- Efectividad en la planeación: Análisis de la proporción entre el tiempo planeado y el real.
- Proporción entre los defectos encontrados y el número de requerimientos funcionales y no funcionales
- Cubrimiento de código usando pruebas unitarias: El porcentaje de código cubierto (ejecutado) cuando las pruebas unitarias son ejecutadas. Estas métricas permiten el seguimiento de qué parte de la aplicación puede ser probada de una manera automática.

3. BIG DATA

En la actualidad, el monto de información almacenada en el mundo se estima alrededor de 1.200 exabytes (10^{18} bytes) [26], los cuales son producidos por múltiples fuentes; por ejemplo, Google procesa más de 24 petabytes (10^{15} bytes) de datos por día y Facebook obtiene más de 10 millones de fotos cada hora [26]. En el mundo de los negocios el 80 % de los datos existen en un formato no estructurado [31] y si este monto fuera solamente libros impresos, ellos cubrirían la superficie entera de Estados Unidos en 52 capas de grueso [26]. Se estima que en algún momento entre 2003 y 2004, el monto de datos creados por el mundo digital aceleró exponencialmente, sobrepasando el monto total de datos creado en los 40.000 años previos de la civilización humana [37].

Teniendo en cuenta estas grandes cantidades de datos que se están generando actualmente, surge el paradigma de Big Data, que es un concepto generalmente usado para describir montos vastos de datos diversos, tanto estructurados como no estructurados, a los cuales las organizaciones pueden acceder de manera rápida, para analizarlos usando herramientas innovadoras, que en conjunto, ayudan a determinar con precisión oportunidades de mejora en la gestión y generación de valor [30].Un problema de Big Data es aquel que cumple con las siguientes tres características[27]:

- Volumen (Escala de los datos)
  Big Data trabaja con grandes cantidades de datos, lo que en la actualidad es algo frecuente teniendo en cuenta que por ejemplo para 2012 cerca de 2.5 exabytes de datos fueron creados cada día y ese número se está duplicando cada 40 meses [27].

- Variedad (Diferentes formas de datos)
  Existen diferentes formas de datos, tanto estructurados como no estructurados, dentro de los que se encuentran: mensajes, actualizaciones, imágenes en redes sociales, lecturas de sensores, señales de GPS desde teléfonos celulares, compras en línea, entre otros [27].

- Velocidad (Análisis de flujo de datos)
  Para muchas aplicaciones, la velocidad de creación de datos es incluso más importante que el volumen. La información en tiempo real, o en tiempos cercanos a ello, hace posible para una compañía ser mucho más ágil que sus competidores y poder tomar decisiones de negocio mucho más informadas y precisas [27].

El cuidado de la salud, en común con muchas otras industrias, está generando grandes cantidades de datos rutinarios, los cuales pueden ser minados e incluso combinados con tweets y blogs. Es un gran reto procesar, analizar

y conservar dicha masa de datos. Darle sentido a esa gran cantidad de información ofrece oportunidades para el mejor tratamiento de una enfermedad, abordar temas de salud pública o para el funcionamiento eficiente de los proveedores de servicios de salud [24]. El análisis de datos contenidos en las historias clínicas electrónicas usando técnicas computacionales es un problema de Big Data [31] debido al gran volumen de información contenida, a la variedad entre datos estructurados y no estructurados, así como también, la velocidad con la que se requieren ciertos análisis. A continuación se presenta un conjunto de aplicaciones existentes de Big Data en el dominio del cuidado de la salud.

3.1. Big Data en Salud

Para identificar los trabajos en el área de la salud que han empleado tecnologías de Big Data, se realizó una revisión bibliográfica entre Febrero y Marzo de 2013 en las bases de datos ISI Web of Science y Scopus, usando los siguientes criterios:

- ISI Web of Sciente
  TI = ("big data" and "health") or TS = ("big data" and "health")
- Scopus
  TITLE-ABS-KEY ("big data" AND health) AND PUBYEAR >1999 AND (LIMIT-TO (DOCTYPE, "ar"))

Cuyos resultados más relevantes se comparan en el cuadro 1. En dicho cuadro se muestra el nombre del artículo a revisar con la respectiva referencia a la sección del presente documento en donde se explica con mayor detalle la tecnología usada, una reseña de la problemática trabajada, una clasificación entre si el artículo propone un aporte teórico o práctico (como por ejemplo el desarrollo de una aplicación) y el tipo de contenido que se trabajaba en el mismo.

Artículo	Tecnología	Problemática	Tipo	Contenido
Uso de Big Data y Cloud Computing para BI operacional en Salud (Sección 3.1.1)	Cloud Computing	Superar la brecha digital para los Sistemas de Información Médica en los países en desarrollo	Práctico	Texto
Big Data en cuidado de la salud personalizado (Sección 3.1.2)	Minería de datos	Perfil de riesgo de enfermedades personalizado	Teórico-Práctico	Texto
Big Data en Ecocardiografía Funcional (Sección 3.1.3)	Cloud computing, robots, Inteligencia Artificial	Poder contar con más variables en la ecocardiografía funcional	Teórico	Imágenes
Adquisición, compresión, encripción y almacenamiento de Big Data en Salud (Sección 3.1.4)	Procesamiento de archivos	Compresión de datos obtenidos en registros electrofisiológicos	Práctico	Imágenes
Big Data en dispositivos médicos (Sección 3.1.5)	Sensores	Confiabilidad y disponibilidad de dispositivos médicos	Teórico	Texto
Desarrollo de un ecosistema de cuidado de la salud usando IPHIs (Sección 3.1.6)	Ontologías	Ecosistema de interoperabilidad en salud para sistemas heterogéneos	Teórico	Texto
Big Data para evaluar el impacto de programas médicos (Sección 3.1.7)	Analytics	Destinar y retener pacientes reales	Teórico	Texto

Cuadro 1: Comparación de artículos de Big Data en salud

A continuación se examinan en detalle cada uno de ellos.

3.1.1. Uso de Big Data y Cloud Computing para BI operacional en Salud

Se propone el uso de Cloud Computing para proveer computación como un servicio utilitario que puede servir de puente para superar la brecha digital para los Sistemas de Información Médica en los países en desarrollo. Se plantea cómo ciertas herramientas pueden explotar las capacidades de la computación en la nube para realizar análisis en Big Data. Usan un Health Management Information System (HMIS) llamado DHIS2 [2] y se analiza cómo sus módulos de Inteligencia de Negocios operacional son usados en los países en vía de desarrollo para gestionar sistemas a nivel de un país o un estado [35].

En los países en vía de desarrollo, donde se tienen sistemas EMR (Electronic Medical Record) pequeños, el manejo de Big Data es poco viable en

[2]Es una herramienta para la recolección, validación, análisis y presentación de datos estadísticos adaptado (pero no limitado) a actividades de la gestión de información médica integrada

términos de costos, por lo cual se propone el uso de AaaS (Analytics as a Service) con el fin de poder manejar grandes cantidades de transacciones de una terminología médica que no es estándar en registros médicos con una inversión razonable. Proponen además implementar terminologías médicas compartidas entre proveedores, para poder así hacer frente al reto de la variedad y con los modelos de nube AaaS manejar fácilmente los problemas de volumen y velocidad [35].

3.1.2. Big Data en cuidado de la salud personalizado

Este proyecto propone un *framework* llamado Collaborative Assessment and Recomedation Engine (CARE) que hace uso de Big Data en un modelo centrado en el paciente para crear un perfil de riesgo de enfermedades personalizado, así como también, un plan de gestión y de bienestar para un individuo usando una técnica de minería de datos llamada filtrado colaborativo. El objetivo de esta técnica es predecir la opinión de un usuario acerca de un ítem o servicio basado en las preferencias conocidas de un gran grupo de usuarios, que es la lógica en las recomendaciones de películas en Netflix.com o de libros en Amazon.com. Para ello se revisan las similitudes en estilo de vida, factores ambientales y predisposiciones genéticas que hacen que ciertas personas tengan mayor probabilidad de padecer enfermedades similares [8].

3.1.3. Big Data en Ecocardiografía Funcional

En este proyecto se hace uso de Ecocardiografía Funcional[3] para el apoyo en el análisis y diagnóstico de enfermedades. Se tiene como particularidad el uso de tecnologías de Big Data, lo cual fué necesario debido a que se decidió ampliar el número de variables a tener en cuenta para el análisis, ya que a menudo es el grupo de factores de deformación, en lugar de uno solo en particular, lo que identifica la presencia fenotípica de una enfermedad [37].

Para ello se propone el uso de las siguientes tecnologías:

1. Eficiencia de datos usando automatización basada en la nube: Uso de almacenamiento en la nube para la gran cantidad de datos obtenidos
2. Uso de robots y automatización en ultrasonidos cardiacos: Mayor eficiencia y eficacia en la toma de exámenes médicos
3. Computación cognitiva: Uso de técnicas de IA
4. Wearable computers: Dispositivos que se pueden llevar como ropa y la forma en que ellos ayudarán a obtener aún más información

[3] También conocida como ultrasonido cardíaco o electrocardiograma

3.1.4. Adquisición, compresión, encripción y almacenamiento de Big Data en Salud

Se propone un formato de compresión (Multiscale Electrophysiology Format) para la gran cantidad de datos obtenidos en registros electrofisiológicos[4] combinado con un sistema de adquisición de información y una base de datos de gran escala SAN (Storage Area Network).

El formato MEF cumple con la HIPAA (Health Insurance Portability and Accountability Act) en términos de protección de la información de cualquier paciente transmitida sobre una red pública. Se habla de la ventaja de usar este formato, ya que gracias a él, los investigadores pueden contar con todos los datos sin tener que eliminar variables por temas de almacenamiento, ya que los archivos quedan en un 20 % de su tamaño original [6].

3.1.5. Big Data en dispositivos médicos

Se utilizan sensores para recolectar información del funcionamiento de los aparatos médicos con el fin de obtener mayor confiabilidad y disponibilidad a bajo costo. Usaron tecnologías de Big Data para predecir el tiempo de vida útil restante, prevenir fallos en el servicio, mantenimientos no planeados y especialmente fallos que pudiesen causar serias pérdidas humanas y económicas [28].

3.1.6. Desarrollo de un ecosistema de cuidado de la salud usando IPHIs

Se propone el desarrollo de un ecosistema de comunicación de Big Data usando Procesadores Intermedios de Información de Salud (IPHI por sus siglas en inglés) que soporten el uso de fuentes de datos heterogéneas. El objetivo es pasar de un modelo de interoperabilidad ineficiente como el de la figura 3 a uno como el de la figura 4. Dichos IPHIs pueden también tener observaciones de pacientes, comentarios de los medios sociales y sus salidas pueden también ir a la prensa o al público. Un ecosistema de información es un ambiente complejo en el cual los datos, los proveedores de información, los usuarios y los procesadores interactúan en un proceso mutuamente interdependiente y transformacional [24].

[4]Electrofisiología: Es el estudio de las propiedades eléctricas de las células y tejidos biológicos.

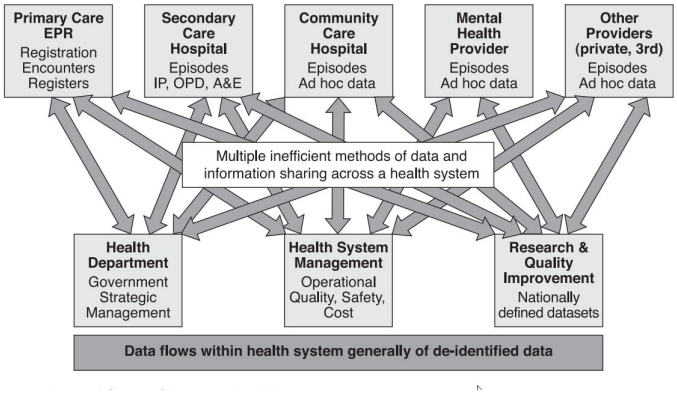


Figura 3: Flujos tradicionales de datos en un sistema de salud (Tomado de: [24])

Los IPHI se encontrarán entre los generadores de la información médica, a menudo los proveedores de servicios de salud y los usuarios de dichos datos. Los usuarios son los gerentes de servicios de salud, comisionistas, generadores de políticas, investigadores, farmacéuticas y otras industrias relacionadas a la salud [24], como se muestra en la figura 4

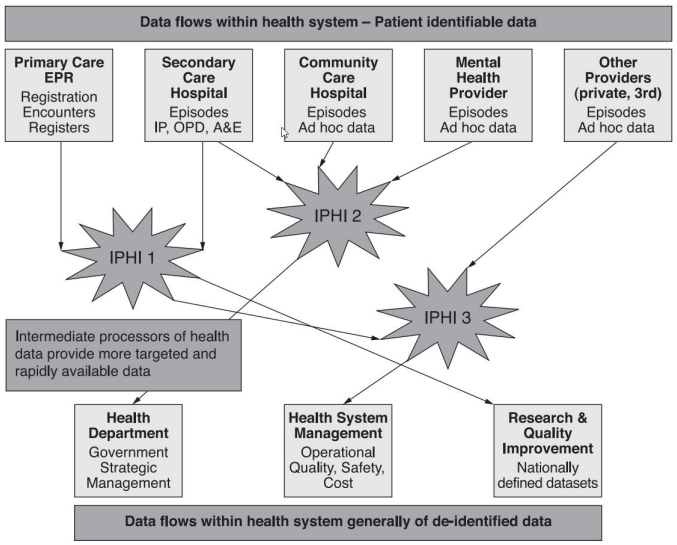


Figura 4: Rol de los IPHI en el sistema de salud (Tomado de: [24])

La implantación de dichos IPHIs en la industria de la salud, a diferencia de otros sectores, necesita hacer prevalecer la privacidad, asegurando a los profesionales, pacientes y el público en general, las disposiciones adecuadas para la gobernanza y seguridad de la información. Un ecosistema de salud de este tipo maximizaría el uso de los datos, creando nuevo conocimiento [24].

3.1.7. Big Data para evaluar el impacto de programas médicos

Se propone la combinación de datos públicos relacionados a un individuo con información vinculada a su condición médica, con el fin de entender de una mejor manera cómo destinar y retener pacientes reales, apalancándose en dichos datos para construir modelos predictivos que pueden asistir de una manera más adecuada a los médicos y realmente impactar en el comportamiento de pacientes con enfermedades crónicas [12]. Dentro de las cosas que se podrían evaluar se tiene:

- Identificar a individuos que tienen mayor probabilidad de ser beneficiados por un tipo particular de programa.
- Identificar individuos que tengan mayor probabilidad de participar activamente en un programa de manejo de enfermedades y qué nivel de divulgación será requerido para asegurar su participación.
- Identificar las razones que impactan más en la adherencia y cumplimiento de un paciente
- Identificar los tipos específicos de divulgación y apoyo que pueden tener mayor impacto en el comportamiento de un individuo y sus resultados.

Finalmente en el cuadro 1 se realiza una comparación de los artículos antes mencionados (los que se identificaron más relevantes de la categoría de aplicaciones de la revisión bibliográfica), encontrando que en su mayoría son propuestas teóricas de usos deseables de las tecnologías de Big Data en el dominio del cuidado de la salud. De igual manera, son pocas las aproximaciones prácticas y palpables del uso de este paradigma en productos o prototipos que realmente hagan realidad los beneficios esperados. También se pudo evidenciar que uno de los factores recurrentes en los artículos, y por lo cual se habla de la necesidad del uso del paradigma Big Data en salud, es la alta cantidad de datos que se requiere analizar para que el personal médico y administrativo pueda tomar las mejores decisiones, entendiendo que una de las mayores fuentes de información es el texto narrativo consignado en las historias clínicas electrónicas. Por lo anterior es importante realizar una revisión de los procesos que pueden ser soportados (Capítulo 4), las tecnologías (Sección 3.2) y herramientas (Sección 3.3) asociadas al paradigma que pueden ser usadas.

3.2. Tecnologías relacionadas a BigData

Como se mencionó anteriormente, cualquier problema de Big Data tiene tres características fundamentales como lo son la velocidad, la variedad y el volumen. Pero dado que Big Data no es como tal una tecnología, sino

más bien un paradigma que debe ser atacado por medio de un conjunto de soluciones, se presentan a continuación (Cuadro 2) algunas de las tecnologías que pueden ayudar a abordar las propiedades de un problema de este tipo:

Tecnología	Sub-tecnología	Velocidad	Variedad	Volumen
NoSQL			X	X
Cloud computing	Infraestructure as a Service (IaaS)	X		X
	Software as a Service (SaaS)	X		X
	Platform as a Service (PaaS)	X		X
	Analytics as a Service (AaaS)	X	X	X
Analytics	Exploratory Data Analysis (EDA)		X	X
	Confirmatory Data Analysis (CDA)		X	X
	Qualitative Data Analysis (QDA)	X	X	X
MapReduce		X		X
Apache Hive		X		X
Apache Pig		X		X

Cuadro 2: Clasificación de tecnologías asociadas a Big Data

3.2.1. Cloud Computing

Es un modelo que permite acceso de red bajo demanda de manera conveniente y ubicua a un conjunto compartido de recursos de cómputo configurables (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con mínimo esfuerzo administrativo o interacción de un proveedor de servicios [35].

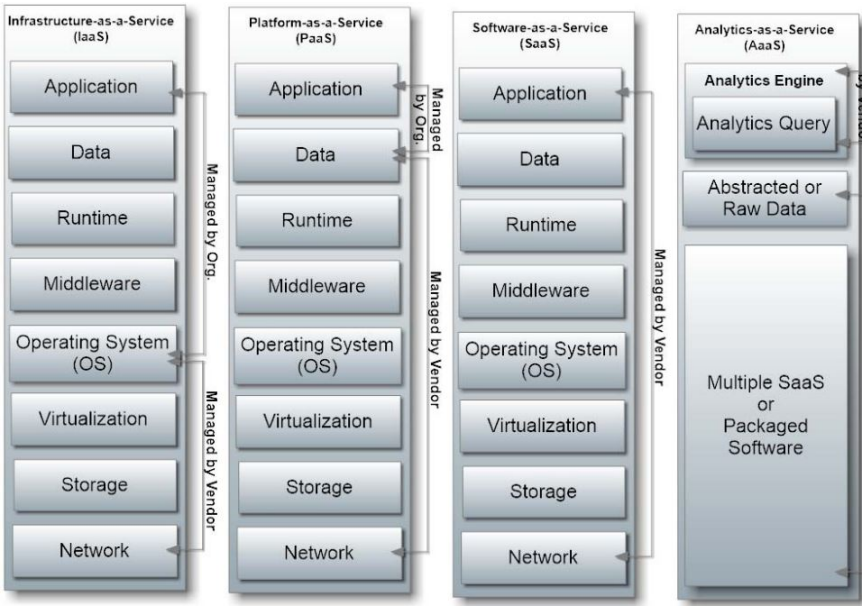


Figura 5: Modelos de servicio Cloud Computing revelantes (Tomado de: [35])

Como se ilustra en la Figura 5, dentro de Cloud Computing se encuentran las siguientes variantes:

- IaaS (Infrastructure as a Service)
  Se basa en el principio de que las inversiones en Hardware se hacen obsoletas y que las organizaciones no tienen que asumir dichos gastos por adelantado, en lugar de eso, las organizaciones rentan nubes de servidores que pueden prestar tiempo de ejecución, middleware, datos y aplicaciones dejando los gastos de mantenimiento al proveedor. Cuando las aplicaciones necesitan más recursos, la infraestructura que la soporta puede crecer tanto como se necesite [35].

- PaaS (Platform as a Service)
  Usualmente representado en los diagramas de Cloud Computing entre las capas SaaS e IaaS, es una amplia colección de aplicaciones de infraestructura (middleware) y servicios (incluyendo plataforma de aplicación, integración, gestión de procesos de negocio y servicios de bases e datos) [13].

- SaaS (Software as a Service)
  Software que es propiedad, distribuido y gestionado remotamente por uno o más proveedores. El proveedor entrega el software basado en un conjunto común de código y de definiciones de datos que es consumido en un modelo de uno-a-muchos por todos los clientes contratados en cualquier tiempo con una base de pago-por-uso o una suscripción basada en uso de métricas [14].

- AaaS (Analytics as a Service)
  El motor de análisis (compuesto por algoritmos y un procesador de consultas) es configurado por el proveedor, mientras que el resto de los datos transaccionales, software requerido, plataforma e infraestructura está en la organización. El proveedor de AaaS generalmente aloja el motor de análisis en IaaS y realiza los análisis de manera rápida, posiblemente co-relacionando con datos externos en un dominio público y entrega los resultados a la organización [35].

3.2.2. Analytics

Es la ciencia de examinar datos en bruto con el propósito de obtener conclusiones acerca de dicha información. Es usado en muchas industrias para permitirle a las compañías y organizaciones tomar mejores decisiones y en ciencias para verificar o desaprobar modelos existentes o teorías. Analytics se distingue de la minería de datos por el alcance, propósito y foco del análisis. Los mineros de datos trabajan con grandes conjuntos de datos usando software sofisticado para identificar patrones no descubiertos y establecer

relaciones ocultas, mientras que en Analytics el foco es en la inferencia, el proceso de llegar a una conclusión, basados solamente en lo que ya es conocido por el investigador [45].

Esta ciencia es generalmente dividida en Análisis Exploratorio de Datos (EDA), donde son descubiertas nuevas características en los datos y Análisis Confirmatorio de los Datos (CDA) donde las hipótesis existentes son probadas como verdaderas o falsas. Existe también el análisis Cualitativo de Datos (QDA) que es usado en las ciencias sociales para llegar a conclusiones basadas en datos no numéricos como palabras, fotografías o videos [45].

3.2.3. MapReduce

Es un estilo de computación que se puede usar para gestionar muchos cálculos de gran escala en una manera que es tolerante a fallos de hardware [36]. Se compone de dos tareas, la primera es la Map, que toma un conjunto de datos y lo convierte en otro donde los elementos individuales se dividen en tuplas y la tarea Reduce que toma la salida de la tarea Map como entrada y combina aquellos datos en un conjunto más pequeño de tuplas [47].

3.2.4. NoSQL

También llamado Not Only SQL, es un enfoque de gestión de datos y diseño de bases de datos que es muy útil para conjuntos de datos muy grandes y que se encuentran distribuidos. Busca resolver los problemas de escalabilidad y rendimiento que trae Big Data y para los cuales las bases de datos relacionales no fueron diseñadas. NoSQL es especialmente útil cuando una empresa necesita acceder y analizar montos masivos de datos no estructurados o que se encuentran almacenados remotamente en múltiples servidores virtuales en la nube. Una base de datos NoSQL puede organizar los datos en objetos, pares llave/valor o tuplas. Contrario a lo que dice su nombre, NoSQL no prohíbe el uso del lenguaje estructurado de consultas (SQL). La base de datos NoSQL más popular es Apache Cassandra, seguido de otras como SimpleDB, Google BigTable, MongoDB, HBase, MemcacheDB y Voldemort [46].

3.2.5. Apache Hive

Apache Hive provee un dialecto basado en SQL, llamado Hive Query Language (HiveQL) para la consulta de datos almacenados en un clúster Hadoop. Hive traduce las consultas a trabajos MapReduce, explotando la escalabilidad de Hadoop, a la vez que presenta una abstracción SQL fácil de aprender. Sin embargo, Hive es mucho más apropiado para aplicaciones de bodegas de datos, donde se analizan datos relativamente estáticos, no

se requieren tiempos de respuesta rápidos y los datos no están cambiando rápidamente [7].

3.2.6. Apache Pig

Apache Pig provee un motor para la ejecución de flujos de datos en paralelo sobre Hadoop y un lenguaje llamado Pig Latin para la expresión de dichos flujos de datos. Dicho lenguaje incluye operadores para muchas de las operaciones de datos tradicionales (join, sort, filter, etc.), como también ofrece la posibilidad de que los usuarios puedan desarrollar sus propias funciones para lectura, procesamiento y escritura de datos. Dado que pig corre sobre Hadoop, hace uso directo de HDFS (Hadoop Distributed File System)[5] y MapReduce (El sistema de procesamiento de Hadoop) [16].

3.3. Distribuciones de Big Data

Teniendo en cuenta las tecnologías asociadas al paradigma de Big Data mencionadas en la sección 3.2, se muestran a continuación algunas de las implementaciones más importantes del mercado:

3.3.1. Hadoop

Es un *framework* que permite el procesamiento distribuido de conjuntos grandes de datos a través de clusters de computadores usando modelos de programación simples. Está diseñado para ir escalando de un solo servidor a miles de máquinas, cada una ofreciendo capacidad de cálculo y almacenamiento. En lugar de confiar en el Hardware para entregar alta disponibilidad, la librería por sí misma está diseñada para detectar y manejar fallos en la capa de aplicación, entregando así un servicio altamente disponible por encima de un clúster de computadores, cada uno de los cuales puede ser propenso a fallos [3].

El *framework* incluye los siguientes módulos:

- Hadoop Common: Los utilitarios comunes que soportan los demás módulos de Hadoop.
- Hadoop Distributed File System (HDFS): Un sistema de archivos distribuido que provee acceso de alto rendimiento a datos de aplicación.
- Hadoop YARN: Un *framework* para la planificación de tareas y gestión de recursos en clúster
- Hadoop MapReduce: Un sistema basado en YARN para el procesamiento en paralelo de grandes conjuntos de datos.

[5]HDFS es un sistema de archivos distribuido que realiza el almacenamiento a través de todos los nodos en un cluster Hadoop

3.3.2. Hortonworks

Hortonworks es un producto de código abierto que se enfoca en llevar al ambiente empresarial, de una manera mucho más transparente, la integración de las tecnologías de Big Data desarrolladas por Apache Software Foundation [19]. A continuación, en la Figura 6, se muestra la arquitectura de Hortonworks, en donde se puede ver la integración de las diversas tecnologías de Apache.

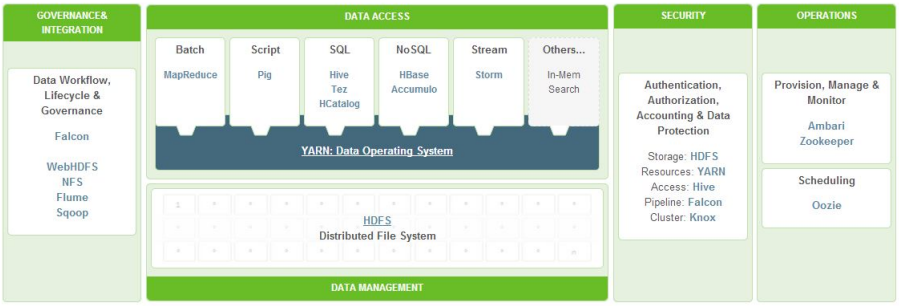


Figura 6: Arquitectura de Hortonworks (Tomado de: [19])

Es una distribución bastante interesante para introducirse en el mundo del procesamiento distribuido con Hadoop, ya que ofrecen máquinas virtuales que pueden ser usadas con VirtualBox o VMWare. Allí en una sola interfaz de usuario se tiene integrado Apache Pig, Apache Hive, HDFS, HBase. Tienen adicionalmente varios tutoriales bastante explicativos del uso de cada una de estas tecnologías. Pero como desventaja se tiene que por su misma naturaleza, no permite comprender de la mejor manera la instalación y configuración de los componentes.

3.3.3. Pivotal Greenplum

Pivotal Greenplum [33] es un producto que propone la integración de tres elementos:

- Una base de datos de procesamiento paralelo para datos estructurados
- Una distribución Hadoop llamada Greenplum HD
- Chorus, una plataforma de productividad que incrementa la agilidad en análisis de equipos globales dedicados a la ciencia de datos. Provee una única interfaz para todos los datos de una organización con bases de datos virtuales, esto con el fin de facilitar la exploración, innovación y colaboración social para el análisis e identificación de oportunidades.
- Le permite a los analistas, personal de TI y ejecutivos participar en análisis Big Data [34].

Siendo la mayor desventaja de este producto que su licencia es de tipo Propietario.

3.3.4. IBM InfoSphere

IBM provee un portafolio de productos para Big Data como son [20]:

- InfoSphere Streams: Permite el análisis continuo de grandes volúmenes de datos en streaming con tiempos de respuesta en sub-milisegundos.
- InfoSphere BigInsights: Una solución basada en Apache Hadoop que permite la gestión y análisis de grandes volúmenes de datos estructurados y no estructurados.
- InfoSphere Data Explorer: Software de descubrimiento y navegación, el cual provee acceso en tiempo real y fusión de Big Data con datos ricos y variados de las aplicaciones empresariales.

3.3.5. Microsoft HDInsight

Hace disponible Apache Hadoop como un servicio en la nube, haciendo el *framework* MapReduce disponible de una manera más simple, escalable y eficiente en términos de costo al soportarse en un ambiente Windows Azure. HDInsight también provee un enfoque eficiente en términos de costo para la gestión y almacenamiento de datos usando Windows Azure Blob [29].

3.3.6. Oracle Big Data

Ofrece una solución completa e integrada para abarcar el espectro completo de los requerimientos empresariales de Big Data. La estrategia en Oracle se centra en poder extender la arquitectura actual de información empresarial para incorporar Big Data, de tal manera que las nuevas tecnologías como Hadoop y Oracle NoSQL puedan trabajar en conjunto con una bodega de datos Oracle [32]. En la figura 7 se presenta la manera en que se orquestan las soluciones de Big Data provistas por Oracle.

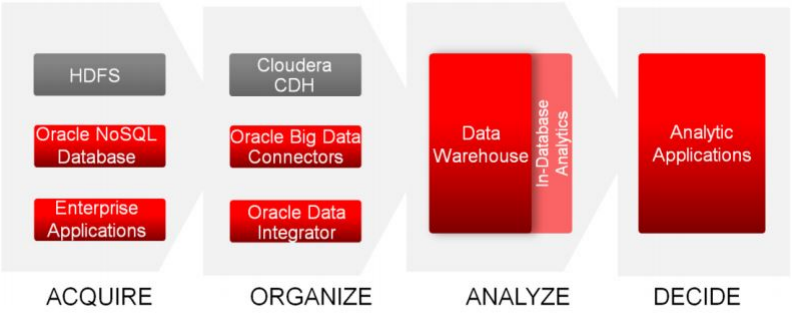


Figura 7: Soluciones de Big Data de Oracle (Tomado de: [32])

En el Cuadro 3 se muestra un resumen comparativo con los diferentes productos relacionados a Big Data:

Tipo	**Producto**	**Base de datos**	**Implementación Hadoop**	**Componente NoSQL**
Open Source	Apache Hadoop	HBase	Hadoop	Hbase
Open Source	Hortonworks	HBase	Hadoop	Hbase
Comercial	Pivotal Greenplum	Greenplum	Greenplum HD	Hbase
Comercial	IBM Infosphere	DB2	InfoSphere BigInsights	HBase
Comercial	Microsoft BigData HD Insights	SQL Server	BigData Solution	SQL Server
Comercial	Oracle Big Data	Oracle	Cloudera	Oracle NoSQL

Cuadro 3: Productos de Big Data

Con el fin de satisfacer el objetivo específico "Seleccionar las técnicas, métodos y herramientas asociados a Big Data que sean relevantes para realizar tareas de pre-procesamiento en minería de texto." en cuanto a lo relacionado a Big Data y teniendo como base las distribuciones (Sección 3.3) y tecnologías (Sección 3.2) antes descritas, a continuación se propone una organización estándar de las mismas en forma de una arquitectura de referencia con un estilo en capas.

3.4. Arquitectura de referencia de Big Data

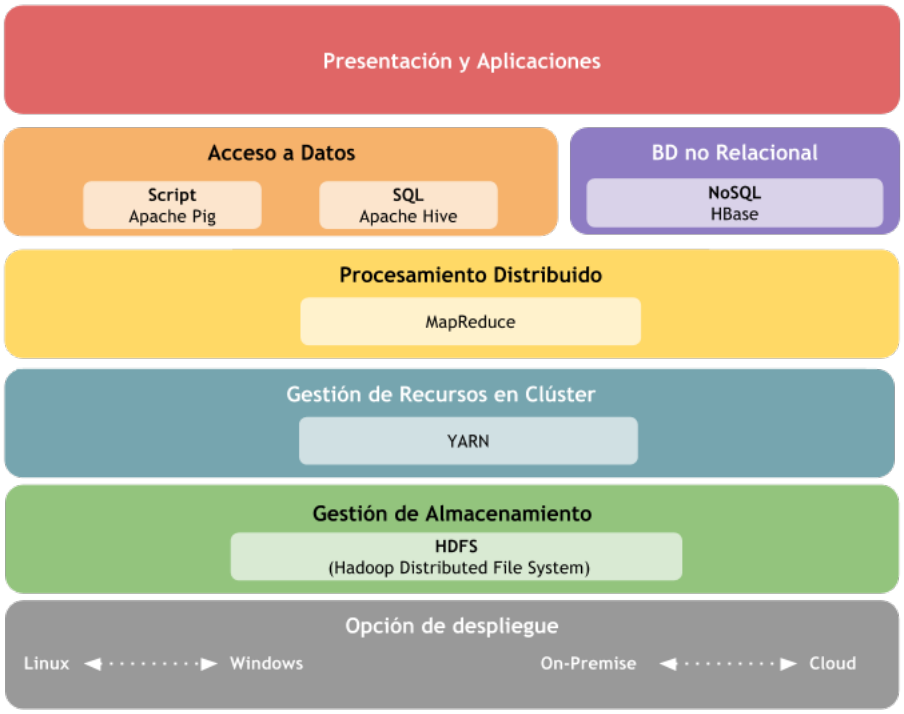


Figura 8: Arquitectura de referencia de Big Data

La arquitectura de referencia propone los siguientes componentes:

1. Opción de despliegue
   Son las diferentes opciones que se tienen para desplegar la arquitectura de Big Data, teniendo por ejemplo sistemas operativos que van desde distribuciones Linux hasta Windows, así como también ambientes de instalación que pueden ser On-Premise (en las instalaciones de la organización) o en la nube.

2. Gestión de almacenamiento
   Es la capa que almacena y procesa los datos de una manera distribuida, es representada por HDFS (Hadoop Distriuted File System). Se encarga de dividir los archivos en pequeñas piezas (llamados bloques) a través de múltiples computadores (Clúster), lo que permite a las funciones map y reduce ser ejecutadas en pequeños subconjuntos de datos. Esto provee la escalabilidad necesaria para el procesamiento de Big Data [19].

3. Gestión de recursos en clúster
   Provee la gestión de recursos para permitir una amplia variedad de métodos de acceso a datos para operar con la información almacenada en HDFS con niveles de servicio y rendimiento predecibles. Esta capa está representada por Apache Hadoop YARN, el cual nace desde la versión 2.0 de Hadoop y su función es separar la gestión de recursos de los componentes de procesamiento [19].

4. Procesamiento distribuido
   Es el componente, que mediante las fases de map y reduce, permite el procesamiento en paralelo.

5. Acceso a datos
   Son los mecanismos mediante los cuales se puede acceder de manera simultánea a un conjunto de datos previamente distribuido. Se tienen entonces varias opciones, dentro de las que se encuentran Apache Pig y Apache Hive. Hive es mucho más adecuado para aplicaciones de bodegas de datos que son implementadas en bases de datos relacionales, reduciendo el impacto de migración y la curva de aprendizaje de las personas que ya manejan el lenguaje SQL. Sus desventajas son que las consultas tienen mayor latencia debido a la sobrecarga que genera el arranque de los trabajos MapReduce. Adicionalmente no es transaccional y no ofrece tareas de actualización, inserción o eliminación a nivel de registros [7]. Por otro lado, Apache Pig ofrece un mayor nivel de abstracción de las tareas MapReduce (map, shuffle y reduce) en forma de un lenguaje de scripting (PigLatin) y a diferencia de HiveQL no es un lenguaje de consultas, sino que le ofrece la oportunidad al usuario de describir exactamente cómo procesar los datos de entrada. Pig Latin ofrece todas las operaciones de procesamiento estándar de MapReduce (join, filter, group by, order, union, etc.) y en general escribir programas en Pig Latin es mucho menos costoso de escribir y mantener que el código Java para MapReduce [16]. Pig además ofrece la posibilidad de crear funciones de procesamiento propias a través de las UDFs (User Defined Functions) que se pueden escribir en Java y en Python.

6. Base de datos no relacional
   Se puede también ver como una forma de acceso a datos pero con la particularidad de que usa HBase[6]. Hay que tener en cuenta que HBase no soporta SQL ya que no es relacional y las aplicaciones son escritas en Java de una manera parecida a MapReduce [19].

7. Presentación y aplicaciones

[6]HBase es un sistema de gestión de base de datos orientado a columnas que se ejecuta sobre HDFS [19]

Es la capa en la que se encuentran las aplicaciones existentes o nuevas que necesitan acceder a las bondades de la arquitectura Big Data propuesta por medio de APIs y servicios como REST.

4. MINERÍA DE TEXTO Y PROCESAMIENTO DE LENGUAJE NATURAL (NLP)

Luego de revisar las tecnologías y herramientas relacionadas a Big Data y teniendo en cuenta que uno de los grandes retos en el dominio del cuidado de la salud es el análisis del texto narrativo consignado en las historias clínicas electrónicas, se pasa a abordar las técnicas de minería de texto y procesamiento de lenguaje natural (NLP por sus siglas en inglés), por ser las más indicadas para el análisis de dicho contenido [11].

La minería de texto puede ser ampliamente definida como un proceso de conocimiento intensivo en el que un usuario interactúa con un grupo de documentos a través del tiempo mediante el uso de un conjunto de herramientas de análisis. De una manera análoga a la minería de datos, la minería de texto pretende extraer información útil a partir de fuentes de datos a través de la identificación y exploración de patrones interesantes. En el caso de la minería de texto, sin embargo, las fuentes de datos son colecciones de documentos, y los patrones interesantes se encuentran no entre los registros de bases de datos sino en los datos de texto no estructurados en los documentos en dichas colecciones [11].

Como se muestra en la figura 9, un proceso simple de minería de texto arranca con un conjunto de documentos de texto, a los cuales se les aplican unas tareas de pre-procesamiento, obteniendo una colección de elementos que son la entrada a las operaciones núcleo de minería de texto cuyo resultado es presentado al usuario final.

Algunas de las tareas más críticas relacionadas a la minería de texto son el pre-procesamiento de colecciones de documentos (categorización de texto, extracción de información, extracción de términos), el almacenamiento de representaciones intermedias, técnicas para analizar dichas representaciones intermedias (como análisis de distribución, clustering, análisis de tendencias y reglas de asociación), así como la visualización de los resultados [11].

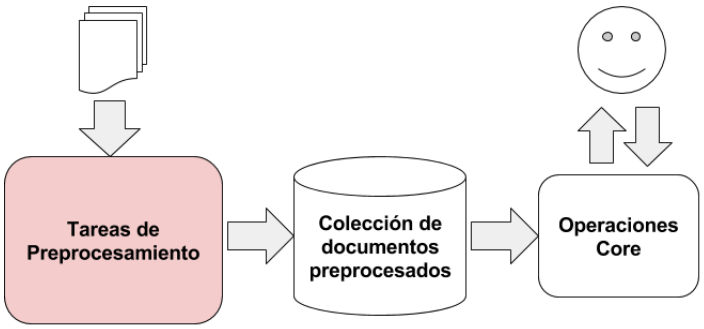


Figura 9: Arquitectura de alto nivel de un sistema de minería de texto (Adaptado de: [11])

Las operaciones de pre-procesamiento se centran en la identificación y extracción de características representativas para documentos en lenguaje natural y son responsables de la transformación de datos no estructurados almacenados en colecciones de documentos a un formato intermedio explícitamente estructurado [11].

Teniendo en cuenta que tanto las tareas de pre-procesamiento como las del núcleo son las dos áreas más críticas para cualquier sistema de minería [11] se encuentra allí una gran oportunidad de optimización de procesos, utilizando tecnologías de Big Data que permitan mejorar los tiempos de respuesta.

4.1. Técnicas de pre-procesamiento en minería de texto

A continuación se presentan algunas de las técnicas más importantes para el preprocesamiento en minería de texto:

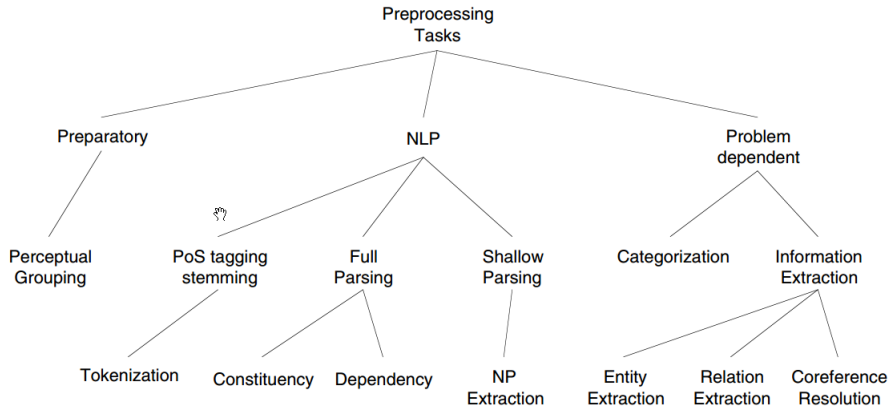


Figura 10: Taxonomía de tareas de preprocesamiento de texto (Tomado de: [11])

4.1.1. Perceptual Grouping

El agrupamiento perceptual se trata de poner piezas juntas en un todo, encontrar regiones con una propiedad uniforme teniendo en cuenta similitud, proximidad, continuidad, simetría y periodicidad [11].

4.1.2. Tokenization

Antes que cualquier otro procesamiento sofisticado, el flujo continuo de caracteres debe ser particionado en componentes significativos. Dicho proceso puede ocurrir a diferentes niveles. Los documentos pueden ser divididos por caractéres, secciones, párrafos, frases, palabras o incluso sílabas y fonemas [11].

4.1.3. Part-of-speech Tagging

El etiquetado POS es la anotación de palabras con la etiqueta apropiada basado en el contexto en el cual ellas aparecen. Las etiquetas POS dividen las palabras en categorías basadas en el rol que ellas juegan en la frase en que aparecen [11]. Proveen información acerca del contenido semántico de una palabra. Los sustantivos denotan por lo general "las cosas tangibles e intangibles", mientras que las preposiciones expresan relaciones entre "cosas". La mayoría de los conjuntos de etiquetas POS hacen uso de las mismas categorías básicas. El conjunto más común de etiquetas contiene siete diferentes: artículo, sustantivo, verbo, adjetivo, preposición, número y nombre propio. Algunos sistemas contienen un conjunto de etiquetas mucho más elaborado. Por ejemplo, el conjunto completo de etiquetas Brown Corpus tiene 87 etiquetas básicas. Por lo general, los etiquetadores POS en alguna etapa de su procesamiento realizan análisis morfológico de las palabras. Por lo tanto, una salida adicional de un etiquetador de este tipo es una secuencia de lemas de las palabras de entrada [11].

4.1.4. Syntactical(Full) Parsing

Los componentes de análisis sintáctico realizan una revisión completa de las frases de acuerdo a cierta teoría gramática. La división básica es entre las gramáticas de dependencia y de constituyentes [11].

- La gramática de constituyentes describe la estructura sintáctica de las oraciones en términos de frases construidas de manera recursiva (secuencias de elementos agrupados sintácticamente). La mayoría de las gramáticas de constituyentes distinguen entre frases de sustantivos, de verbos, preposicionales, de adjetivos y cláusulas. Cada oración puede consistir en frases más pequeñas o simplemente palabras de acuerdo a las reglas de la gramática. Adicionalmente, la estructura sintáctica de oraciones incluye los roles de las diferentes frases. Por lo tanto, una oración de sustantivo puede ser etiquetada como el sujeto de una oración, su objeto directo o el complemento [11].

- La gramática de dependencias, por otro lado, no reconoce los constituyentes como unidades linguísticas separadas sino que se enfoca en las relaciones directas entre frases. Por ejemplo, un sujeto y sustantivos directos de una oración típica dependen del verbo principal, un adjetivo depende del sustantivo que modifica y así sucesivamente [11].

4.1.5. Shallow Parsing

El análisis eficiente y preciso de texto sin restricciones no se encuentra dentro del alcance de las técnicas actuales. Los algoritmos estándar son muy

costosos para el uso en corpus[7] muy grandes y no son lo suficientemente robustos. El análisis sintáctico superficial (Shallow Parsing) compromete velocidad y robustez en el procesamiento al sacrificar la profundidad del análisis. En lugar de proveer un análisis completo de toda una frase, se producen sólo partes que son fáciles y sin ambigüedades. Típicamente, frases pequeñas de sustantivos y verbos son generadas (**Noun Phrase Extraction**), mientras que las cláusulas complejas no son formadas. Del mismo modo, se podrían formar la mayoría de las dependencias importantes, pero las poco claras y ambiguas se dejan sin resolver. Para los fines de la extracción de información, el análisis sintáctico superficial es por lo general suficiente y por lo tanto preferible al análisis completo debido a su superioridad en velocidad y robustez [11].

4.1.6. Categorización

Las tareas de categorización de texto (a veces llamado clasificación de texto) etiquetan cada documento con un número pequeño de conceptos o palabras clave. El conjunto de todos los posibles conceptos o palabras clave es normalmente preparado manualmente, cerrado y comparativamente pequeño. La relación de jerarquía entre las palabras clave también es preparada de forma manual [11].

4.1.7. Extracción de informacion

La Extracción de información es quizás la técnica más prominente actualmente usada en las operaciones de preprocesamiento en minería de texto. Sin dicha técnica, los sistemas de minería de texto podrían tener unas capacidades de descubrimiento de conocimiento mucho más limitadas. Esta técnica debe ser distinguida de la recuperación de información (IR por sus siglas en inglés) o lo que se llama de manera informal "Búsqueda" ya que IR (Information Retrieval) retorna documentos que coinciden con una consulta dada pero aún requiere que el usuario lea dichos documentos para localizar la información relevante. La Extracción de Información (EI), por otro lado, tiene como objetivo la localización de información relevante y presentarla en un formato estructurado (típicamente en un formato tabular). Para los analistas y otros trabajadores del conocimiento, EI puede ahorrar tiempo valioso al acelerar de manera considerable el trabajo de descubrimiento [11].

La extracción de información se puede dividir en:

- Extracción de entidades: Identificar todos los lugares, personas, organizaciones, fechas, monedas, porcentajes, etc.
- Extracción de relaciones: Detección y clasificación de relaciones semánticas entre expresiones.

[7] Colección de documentos

- Resolución de correferencias: Anáfora o resolución de correferencias es el proceso de hacer coincidir los pares de expresiones de PLN (Procesamiento de Lenguaje Natural) que se refieren a la misma entidad en el mundo real.

Tanto las tareas de categorización de texto como de Extracción de Información le permite a los usuarios ir de una representación de los documentos leible por una máquina a una entendible por una máquina como se muestra en la figura 11.

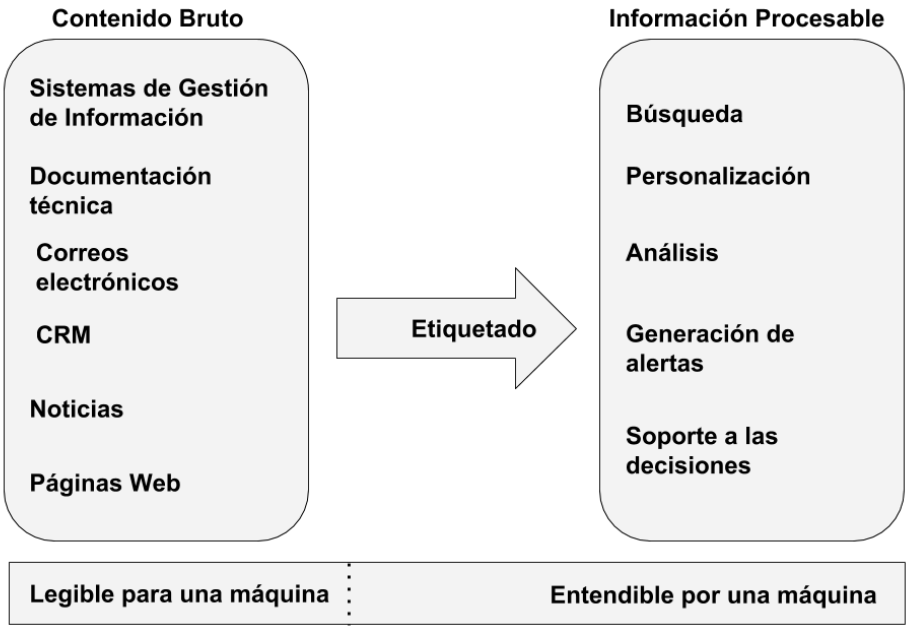


Figura 11: Cerrar la brecha entre los datos brutos e información procesable (Tomado de: [11])

A continuación se presentan los hallazgos en cuanto a la aplicación de Big Data en el dominio de la minería de texto y procesamiento de lenguaje natural (NLP por sus siglas en inglés).

4.2. Aplicaciones de Big Data en Minería de Texto

Luego de realizar búsquedas en las bases de datos científicas ISI Web Of Knowledge y Scopus acerca de aplicaciones específicas de Big Data dirigidas a minería de texto y el procesamiento de lenguaje natural, se seleccionaron dos artículos relevantes que se presentan a continuación:

4.2.1. Big Data para análisis de datos no estructurados

Se propone la utilización de Big Data unido a minería de texto para trabajar con contenido narrativo en forma de tweets públicos, usando tecnologías como HBase, Java, REST y Hadoop [9].

Para ello se establecieron las siguientes fases:

1. El establecimiento de la conexión, seguida por la transmisión de los tweets públicos de Twitter utilizando Java (los datos se recuperan a partir de análisis de respuesta XML).

2. La construcción de un Hbase y almacenamiento de los datos en él, después del análisis de sentimiento (toda la comunicación se realiza a través de llamadas REST).

3. Construcción de un Front-end para que el cliente interactúe con la Hbase a través del *framework* Java para obtener los datos pertinentes requeridos

Dado que es un proyecto en progreso, solamente se ha completado la primera fase, en donde se toman los tweets y los colocan en la base de datos HBase, pero no se realiza minería de texto aún.

4.2.2. GATECloud.net: una plataforma de código abierto para el procesamiento de texto a gran escala en la nube

Se mencionan varias estrategias para mejorar los tiempos de respuesta en procesamiento de lenguaje natural, entre las cuales se tenían dos opciones: el uso de MapReduce y el manejo de tecnologías de Cloud Computing, siendo para ellos mejor la segunda, bajo un modelo PaaS (Platform as a Service), ya que no es necesario reescribir los algoritmos de procesamiento de texto que se van a ejecutar. El nombre de la solución es GATECloud.net, que es una plataforma Web en la que los científicos pueden realizar sus experimentos de minería de texto, cuyos servicios son cobrados mediante el modelo pago por uso y donde no se proponen librerías (APIs) para el empleo de la herramienta desde otros sistemas [39].

4.3. Herramientas de Minería de texto

A continuación se enuncia un conjunto de herramientas que son usadas para minería de texto:

4.3.1. Gate

Gate es un software de código abierto capaz de resolver problemas de procesamiento de texto, contando con una comunidad madura y extensa de

desarrolladores, usuarios, educadores, estudiantes y científicos. Provee un proceso definido y repetible para la creación de flujos de trabajo para el procesamiento de texto, de tal forma que dichos flujos puedan ser mantenibles y robustos [15].

4.3.2. Weka

Weka es un proyecto desarrollado en la Universidad de Waikato en Nueva Zelanda y construido en Java que se compone de una colección de algoritmos de aprendizaje de máquina y diversas tareas de minería, los cuales pueden ser aplicados directamente a un conjunto de datos por medio de una interfaz gráfica o ser llamados como librería desde una aplicación Java. Weka contiene tareas de pre-procesamiento de datos, clasificación, regresión, clustering, reglas de asociación y visualización. Weka es software libre publicado bajo la Licencia Pública General GNU [43].

4.3.3. Stanford CoreNLP

Stanford CoreNLP proporciona un conjunto de herramientas de análisis de lenguaje natural, las cuales toman como entrada texto sin procesar y dan como resultado formas base de las palabras, sus partes de la oración, si son nombres de empresas, personas, etc., se normalizan fechas, horas y cantidades numéricas, y marcan la estructura de las oraciones en términos de frases y dependencias de palabras, indican qué sustantivos se refieren a las mismas entidades, indican sentimientos, etc. Stanford CoreNLP es un marco integrado. Su objetivo es hacer que sea muy fácil de aplicar gran cantidad de herramientas de análisis lingüístico a un fragmento de texto. A partir de texto plano, puede ejecutar todas las herramientas en él con sólo dos líneas de código. Está diseñado para ser altamente flexible y extensible. Con una única opción que puede cambiar qué herramientas deben estar habilitadas y cuáles deben ser desactivadas. Sus análisis proporcionan los bloques de construcción fundamentales para el nivel superior y aplicaciones de comprensión de texto específicos de dominio.

Stanford CoreNLP integra muchas de las herramientas desarrolladas por el grupo de procesamiento de lenguaje natural realizados en la Universidad de Stanford, incluyendo Part of Speech (POS), el Named Entity Recognition (NER), el analizador, el sistema de resolución de la correferencias, el análisis de sentimientos, y las herramientas del patrón bootstrapped de aprendizaje. La distribución básica proporciona archivos modelo para el análisis en idioma Inglés, pero el motor es compatible con los modelos para otros idiomas. Stanford CoreNLP está escrito en Java y es licenciado bajo la GNU General Public License [25].

4.3.4. KNIME Text Processing

Permite leer, procesar, minar y visualizar datos textuales en una manera conveniente. Provee funcionalidades para [21]

- Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés)
- Minería de texto
- Recuperación de información

4.3.5. Apache UIMA

UIMA (Unstructured Information Management Architecture) es un estándar OASIS desde marzo de 2009, para el análisis de grandes volúmenes de información no estructurada con el fin de descubrir conocimiento que es relevante a un usuario final. Adicional a ello, provee capacidades para envolver componentes como servicios de red y puede ser escalado a grandes volúmenes por medio de replicación de *pipelines* de procesamiento sobre un clúster de nodos interconectados. Los componentes son desarrollados tanto en Java como en C++ [1].

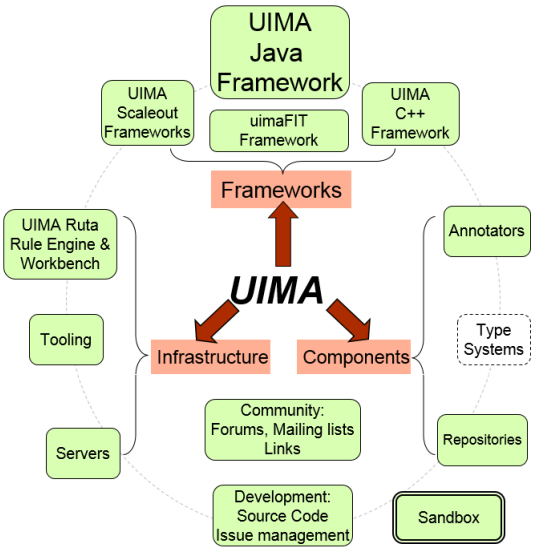


Figura 12: Arquitectura UIMA (Tomado de: [1])

4.3.6. SAS Enterprise Miner

El software SAS (Statistical Analysis System) Enterprise Miner (SAS Institute) es utilizado para el análisis de texto que incluye conversión de texto no estructurado en datos estructurados, clustering de las entradas

de documentos/datos y visualización de enlaces entre conceptos. Utiliza la técnica TF-IDF para identificar algunos de los conceptos o términos importantes. SAS es un Software de tipo propietario desarrollado en la North Carolina State University [17].

Proyecto	**Licencia**	**Versiones**	**Lenguaje**	**Facilidad de uso para programación**
Gate	Open Source	Interfaz de usuario y API	Java	Necesidad de creación de un pipeline para su uso
Weka	Open Source	Interfaz de usuario y API	Java	Pocas funciones de pre-procesamiento
Stanford CoreNLP	Open Source	API	Java	Gran variedad de funciones de pre-procesamiento
KNIME	Open Source	Interfaz de usuario	Java	No ofrece API
Apache UIMA	Open Source	API	Java y C++	Necesidad de creación de un pipeline para su uso
SAS Enterprise Miner	Propietario	Interfaz de usuario	C	N/A

Cuadro 4: Comparación y análisis de herramientas de minería de texto y NLP

Finalmente, como se muestra en el Cuadro 4 dentro de las herramientas de minería de datos y procesamiento de lenguaje natural expuestas anteriormente, se realizaron pruebas con aquellas que fuesen de código abierto, que ofrecieran un API, que fuesen desarrolladas en Java y que ofrecieran una mayor facilidad a la hora de programar funciones de pre-procesamiento, habiendo seleccionado dos de ellas como las más adecuadas para el sistema: Weka y Stanford CoreNLP, siendo el segundo el que mayores funcionalidades de pre-procesamiento y facilidad de programación ofrecía.

5. BIGTEXTS

Teniendo como partida la pregunta de investigación ¿Cómo generar un framework que ofrezca funcionalidades pre-construidas para ejecutar tareas de pre-procesamiento en minería de texto basado en tecnologías de Big Data, en donde se logre adicionar un nivel de abstracción a la complejidad que implica tener que enlazar las diferentes tecnologías disponibles y así poder mejorar los tiempos de procesamiento en sistemas de minería de texto?, se propone BigTexts, un modelo de aplicación de Big Data que soporta la fase de pre-procesamiento de manera paralela, haciéndola más rápida y eficiente.

BigTexts es un framework para pre-procesamiento de datos en minería de texto basado en tecnologías de Big Data, el cual ofrece funcionalidades pre-construidas que pueden ser usadas mediante la ejecución de una aplicación independiente (con una interfaz gráfica de usuario - GUI) o como librería (API) de una aplicación existente.

5.1. BigTexts en la arquitectura de referencia

Para el cumplimiento del objetivo específico "Seleccionar las técnicas, métodos y herramientas asociados a Big Data que sean relevantes para realizar tareas de pre-procesamiento en minería de texto." en cuanto a las herramientas de Big Data, a continuación (Figura 13) se presentan resaltadas en rojo las tecnologías seleccionadas.

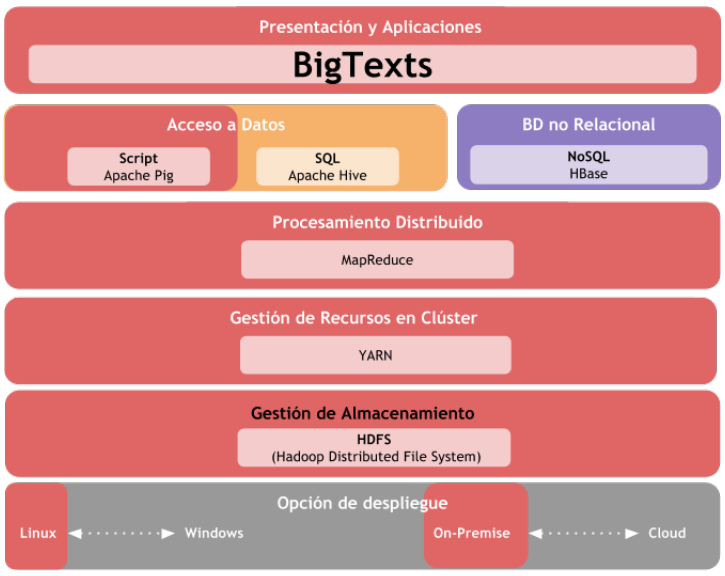


Figura 13: BigTexts en la arquitectura de referencia

Teniendo en cuenta la arquitectura de referencia de Big Data expuesta

en la Sección 3.4 BigTexts se encuentra en la capa superior (Presentación y Aplicaciones), usando Apache Pig como acceso a datos por su facilidad para construir funciones propias en el lenguaje Java. No se usa la base de datos no relacional HBase, ya que el sistema sólo realiza el pre-procesamiento y no almacena los documentos. MapReduce para el procesamiento distribuido, YARN como gestor de recursos en clúster y HDFS para el almacenamiento distribuido. Se utiliza como opción de despliegue el sistema operativo Linux, específicamente Ubuntu ya que no requiere capas intermedias como Cygwin[8] para la instalación de los componentes, y una estrategia On-premise y no una Cloud computing por restricciones económicas.

5.2. Generalidades de Big Texts

En términos generales, como se muestra en la Figura 14, BigTexts está compuesto por dos artefactos, un cliente (Cliente BigTexts) y un servidor (BigTexts).

El cliente ofrece una interfaz gráfica por medio de la cual el usuario puede seleccionar los archivos a procesar, las tareas de pre-procesamiento a ejecutar (con sus respectivas parametrizaciones) y la forma de entrega[9]. Dicha aplicación cliente también puede ser usada por una aplicación externa como librería. El cliente se encarga de enviarle los archivos al servidor mediante un FTP y los llamados para las ejecuciones se realizan de manera asíncrona, enviando mensajes en formato XML[10] a un servidor de colas.

El servidor BigTexts se conecta al servidor de colas y obtiene el mensaje encolado, transformándolo de XML a objetos. Posteriormente el mensaje convertido en objetos es ejecutado en el clúster Hadoop mediante la construcción de un script de Apache Pig que usa UDFs[11] con las tareas de pre-procesamiento. El clúster Hadoop está compuesto por una máquina principal (el master), la cual se encarga de gestionar tanto el almacenamiento como el processamiento en paralelo, y una serie de máquinas secundarias (slaves) que son las que almacenan los bloques de datos y los procesan de manera paralela. Finalmente se entrega un conjunto de documentos pre-procesados en el HDFS o en el directorio FTP, según lo haya seleccionado el usuario.

Se usa además una base de datos PostgreSQL para el almacenamiento

[8] Colección de herramientas GNU y Open Source que proveen funcionalidad similar a una distribución Linux en Windows

[9] La forma en la que el cliente requiere que se le entreguen los documentos resultado del proceso, que puede ser en un directorio del HDFS o en el servidor FTP.

[10] Se escoge XML sobre otros lenguajes como JSON o YAML, ya que permite validación usando XSD, transformaciones usando XSLT, puede recibir texto marcado en las parametrizaciones y se podía utilizar las funcionalidades nativas de Java (Usando JAXB que ya se encuentra disponible en el JDK) y no utilizar librerías adicionales como por ejemplo GSON para JSON

[11] User Defined Functions, La forma en que Apache Pig permite la creación de funciones personalizadas en Java.

de la siguiente información de auditoría relacionada a las ejecuciones:

- Fecha y hora de inicio de procesamiento
- Fecha y hora final de procesamiento
- Fecha y hora de carga de archivos
- Nombre y peso de archivos pre-procesados
- Las tareas de pre-procesamiento parametrizadas y
- El número de máquinas secundarias disponibles

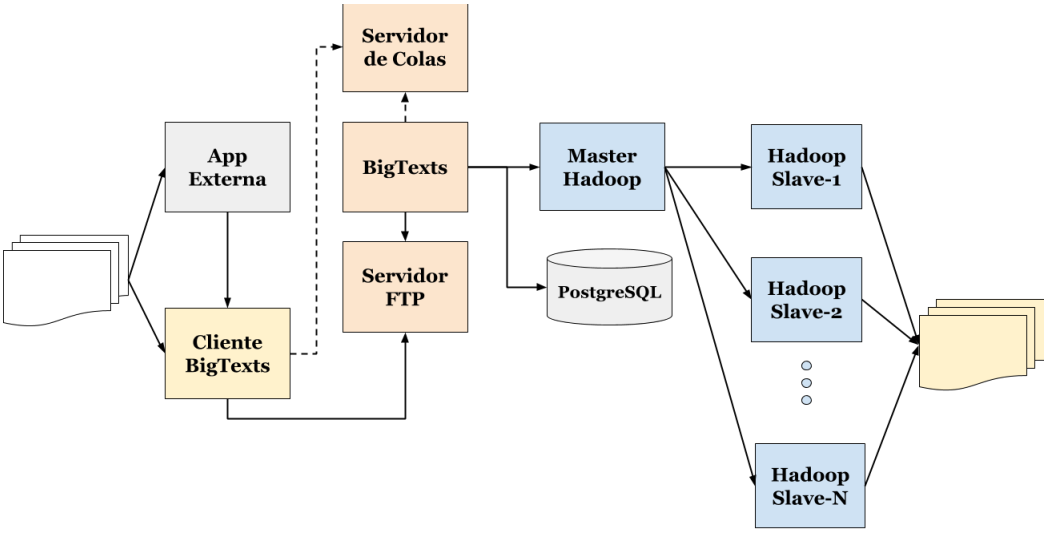


Figura 14: BigTexts

A continuación se presentan las principales características ofrecidas por el *framework*.

5.3. Características

Como se muestra en la figura 15, en la situación actual se cuenta con un conjunto de documentos, a los cuales le es aplicada una serie de tareas de pre-procesamiento de manera secuencial, para obtener una colección de documentos pre-procesados. La situación deseada, apoyada por BigTexts, toma los documentos iniciales, los particiona en el cluster Hadoop mediante HDFS, y usando MapReduce, ejecuta las tareas de pre-procesamiento en paralelo, llegando finalmente a la colección de documentos pre-procesados.

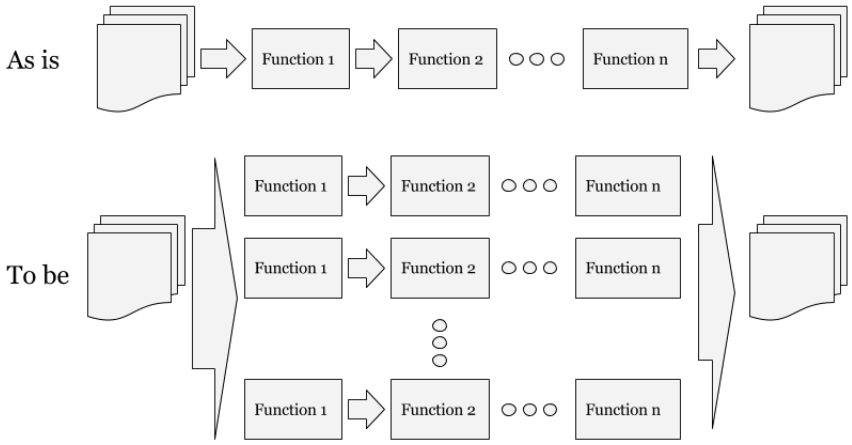


Figura 15: Situación actual vs situación deseada

BigTexts ofrece las siguientes funcionalidades de pre-procesamiento en minería de texto, las cuales fueron construidas usando mayoritariamente el software Stanford CoreNLP mencionado en la sección 4.3.3 encapsulado en UDFs de Apache Pig:

- Identificación de correferencias[12] en inglés
- Lematización[13] usando el algoritmo Lovins Stemmer para inglés
- Named Entity Recognition[14] en inglés
- Named Entity Recognition en español
- Part Of Speech[15] en inglés
- Part Of Speech en español
- Named Entity Recognition basado en expresiones regulares
- Lematización usando el algoritmo Snowball Stemmer para inglés
- Lematización usando el algoritmo Snowball Stemmer para español
- Partición de textos por frases
- Tokenización

[12]Es la tarea de encontrar todas las expresiones que se refieren a la misma entidad en un texto [40]
[13]Reducir una palabra a su lema o raíz[23]
[14]Etiquetar palabras en un texto como verbos, adjetivos, sustantivos, etc. [42]
[15]Etiquetar palabras en un texto como personas, nombres de compañías, géneros, paises, ciudades, etc.[41]

- Paso a mayúsculas de un texto

La adición de una nueva tarea de pre-procesamiento ha sido diseñada para ser un proceso sencillo y flexible. Para ello simplemente se crea una nueva clase en la cual se implementa la funcionalidad y se adiciona al catálogo de tareas de pre-procesamiento[16].

Si se requiere identificar la información relacionada a un registro en particular, lo que se debe hacer es diferenciar dicho registro en un archivo separado, para que de esta manera el resultado de las tareas de preprocesamiento para ese registro en particular sea fácilmente referenciable.

A continuación se presenta una vista general de la arquitectura de BigTexts, para mayor información revisar el Anexo 4, Arquitectura BigTexts.

5.4. Arquitectura de BigTexts

Este informe presenta la arquitectura propuesta para desarrollar el proyecto BigTexts: Framework de pre-procesamiento de datos en minería de texto basado en tecnologías de Big Data, estableciendo primero los atributos de calidad, luego se define el estilo arquitectónico selecionado y tres de las vistas del sistema, para mayor información revisar el Anexo 4. Arquitectura BigTexts.

5.4.1. Atributos de Calidad

Los siguientes son los atributos y subatributos de calidad identificados según la clasificación de atributos de calidad de la norma ISO 9126 [38], que se extrajeron y propusieron a partir de los requerimientos.

Prioridad	Atributo	Sub-atributo	Concerns
5	Eficiency	Time behavior	Dado un conjunto n de textos, el tiempo de preprocesamiento será igual o menor al tiempo de preprocesamiento secuencial, dividido en el número de máquinas disponibles en el clúster. Donde las máquinas disponibles en el clúster son al menos tan robustas como la máquina inicial
4	Functionality	Interoperability	El Framework debe ser usado por otros sistemas en forma de API o librería
3	Usability	Understandability	El framework debe usar los patrones de usabilidad Cancel, Alerts, Status indication, Tooltips, Help y Undo
2	Reliability	Recoverability	El framework debe ser capaz de volver a lanzar procesos fallidos
1	Maintainability	Changeability	El sistema debe tener la capacidad de fácilmente agregar un nuevo algoritmo de pre-procesamiento

Cuadro 5: Atributos de calidad

5.4.2. Estilo Arquitectónico

Para seleccionar los estilos arquitectónicos relevantes para los atributos de calidad identificados se utilizó la siguiente tabla donde se califica cada

[16]Un archivo XML con todas las tareas de preprocesamiento y los valores de parametrización por defecto.

atributo de calidad (con su respectiva prioridad) en comparación con los estilos arquitectónicos tanto estructurales como para sistemas interactivos. Las calificaciones se basaron en el artículo *A method of selecting appropriate software architecture styles:Quality Attributes and Analytic Hierarchy Process* [44], de la siguiente manera:

- 2: El estilo arquitectónico tiene impacto MUY positivo sobre un atributo de calidad.
- 1: El estilo arquitectónico tiene impacto positivo sobre un atributo de calidad.
- 0: Representa que no tiene impacto, es neutral o no especificado.
- -1: El estilo arquitectónico tiene impacto negativo sobre un atributo de calidad.
- -2: El estilo arquitectónico tiene impacto MUY negativo sobre un atributo de calidad.

		Eficiency (Time behavior)	**Functionality (Interoperability)**	**Usability (Understandability)**	**Reliability (Recoverability)**	**Maintainability (Changeability)**	**Total**
		5	4	3	2	1	
Estructurales	**Cliente / Servidor**	2	0	0	0	-1	**9**
	Pipes and filters	0	0	-2	0	1	**-5**
	Layers	-2	0	0	0	2	**-8**
Sistemas interactivos	**Model-View-Controller**	0	0	1	0	1	**4**
	Presentation - Abstraction - Control	-1	0	1	0	1	**-1**
Comunicación	**Publish - Subscribe**	-1	2	0	2	0	7
	Forwarder - receiver	-1	0	0	0	-1	**-6**

Cuadro 6: Tabla de escogencia de estilos arquitectónicos

Se realiza la suma ponderada por cada estilo arquitectónico obteniendo unos totales para cada uno de ellos[17], llegando a un estilo arquitectónico estructural, uno de sistemas interactivos y uno de comunicación. Dichos estilos se definen a continuación:

Cliente / Servidor: Se escoge este estilo arquitectónico debido a que existe la posibilidad de contar con varios clientes y además se tiene la posibilidad de distribuir el procesamiento en un clúster de servidores. Se promueve

[17]Para el estilo arquitectónico Cliente / Servidor en el atributo de calidad *Eficiency (Time behavior)* se establece un valor de 1, dado que se pretende usar componentes relacionados a Big Data como Hadoop que permitan tener mejores tiempos de respuesta en clústers de servidores.

mediante este patrón la modificabilidad y el reuso, ya que se crean servicios comunes, los cuales al existir una necesidad de modificación, disminuye el impacto, ya que dicho cambio solamente se tiene que hacer en un solo sitio, o en un número pequeño de componentes. Se mejora la escalabilidad y la disponibilidad al centralizar el control de esos recursos y servicios, mientras que se pueden distribuir dichos recursos en múltiples servidores físicos.

Model-View-Controller: es el patrón que determina el comportamiento y la interacción con el usuario, es decir, describe cómo funcionará el componente cliente del patrón dominante. Es muy útil dado que la interfaz de un sistema de software es normalmente la porción más modificada e interactiva de la aplicación.

Publish - Subscribe: Es el que permitirá el llamado asíncrono entre el cliente y el servidor, garantizando la extensibilidad y confiabilidad del sistema, ya que a través de su esquema de colas permitirá que los nodos se comuniquen asincrónicamente. Es la manera de crear mecanismos de integración que soportan la habilidad de transmitir mensajes entre productores y consumidores de manera tal que no conocen la identidad del otro o probablemente su existencia. Es decir, se logra bajo acoplamiento entre los clientes y el servidor. Por consiguiente el número de componentes, interfaces y conexiones es: añadido, eliminado y modificado, junto con una caracterización de la complejidad de los cambios, cancelaciones, modificaciones.

Para describir la arquitectura se utilizará el modelo de Kruchten [22]; este modelo consiste en 5 vistas las cuales se explican a continuación:

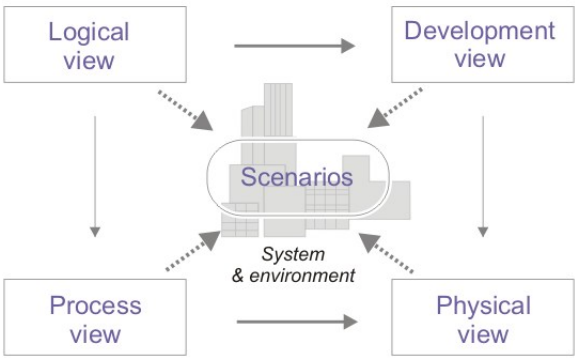


Figura 16: Modelo 4+1 de Kruchten (Tomado de: [22])

5.4.3. Vista de Escenarios

Esta vista permite la descripción de la arquitectura usando un conjunto de casos de uso o escenarios, siendo un punto de partida para el diseño. Se identifican las interacciones entre los usuarios y los requerimientos del sistema[22].

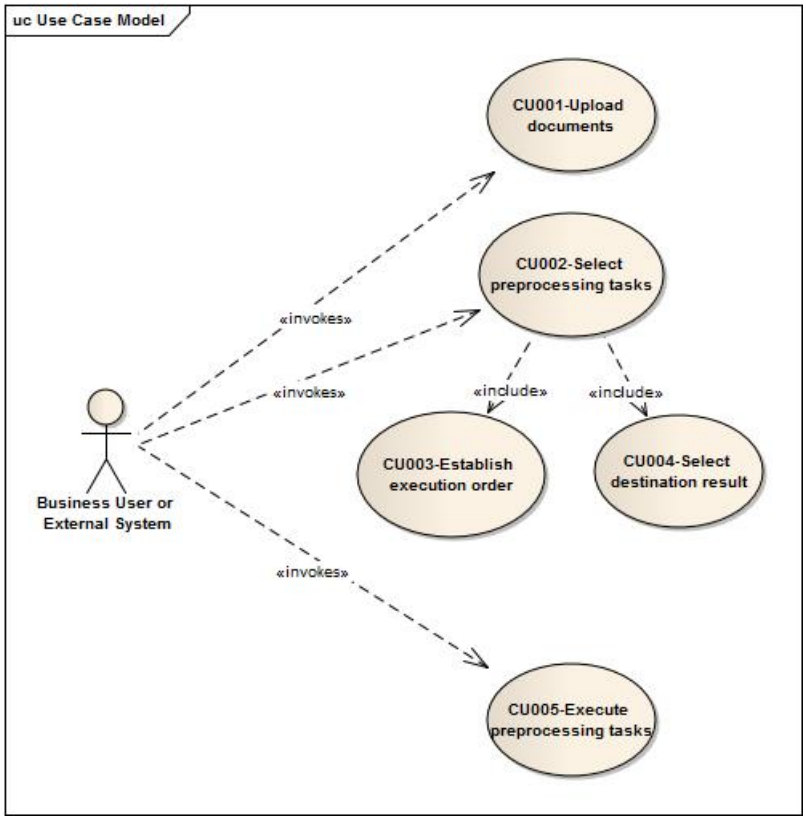


Figura 17: Vista de Escenarios

1. CU001-Upload documents: Describe el proceso de realizar el cargue de documentos para su posterior procesamiento

2. CU002-Select preprocessing tasks: Describe la selección de las tareas de pre-procesamiento que se desean realizar en el sistema. Las tareas que se podrán seleccionar son Part Of Speech, Splitter, Tokenizer, Entity Recognition, búsqueda basado en expresiones regulares (NE Transducer) e identificación de coreferencias

3. CU003-Establish execution order: Se establece el orden en que se deben ejecutar las tareas seleccionadas y configuradas en el CU002.

4. CU004-Select destination result: El sistema elige la forma en que quiere que le sea entregado el resultado, teniendo como opciones: una ubicación en un directorio FTP, en el sistema de archivos local, o en un directorio HDFS.

5. CU005-Execute preprocessing tasks: El sistema ejecuta la lista de tareas de pre-procesamiento sobre los documentos cargados.

Para mayor información acerca de los casos de uso revisar el Anexo 1. Entrevista de requerimientos y el Anexo 2. Documento de casos de uso.

5.4.4. Vista Lógica

Esta vista permite la identificación de los elementos estructurales del sistema y el estilo arquitectónico seleccionado[22].

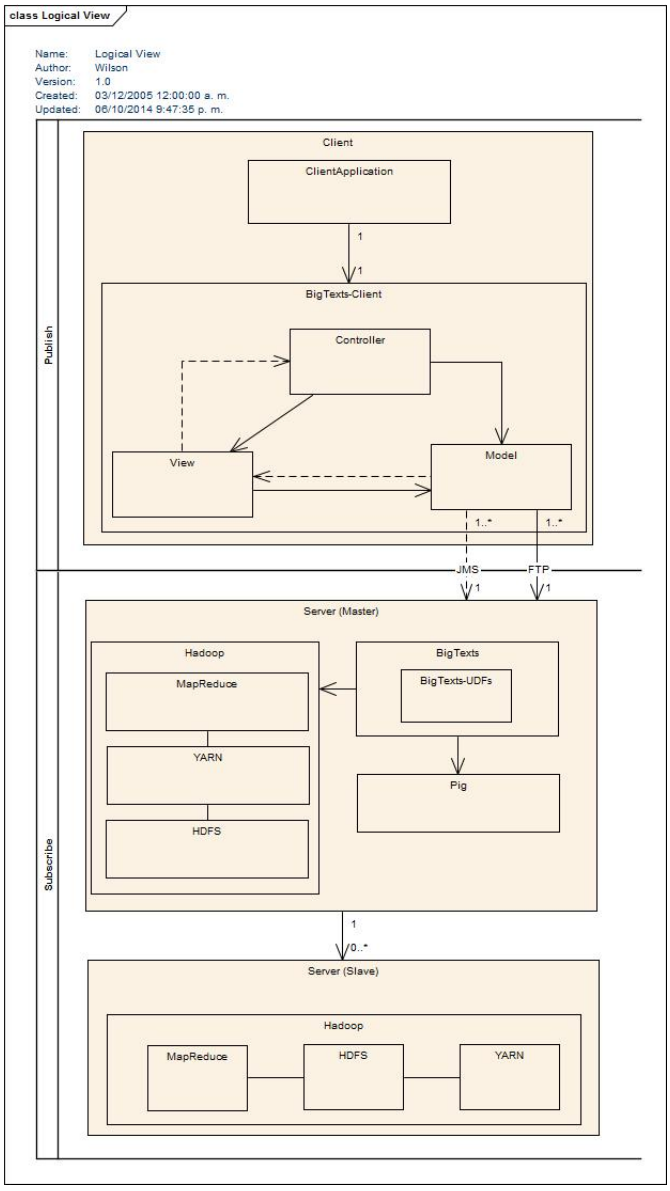


Figura 18: Vista Lógica

1. Client: Es la máquina cliente en donde se encuentra tanto el cliente de

BigTexts, como la aplicación externa que lo utiliza.

   a) ClientApplication: Es el sistema externo que usa el API de BigTexts.
   b) BigTexts-Client: Es la aplicación de escritorio que es usada tanto por un usuario de negocio (por medio de una interfaz gráfica), como por un sistema externo (como librería que expone un API). Dentro de este cliente se encuentra implementado el patrón MVC. Es el componente que invoca los servicios del componente servidor. Se utiliza el patrón MVC para mantener separada la funcionalidad de interfaz de la funcionalidad de aplicación.
      1) Model: Contiene la funcionalidad principal y los datos.
      2) View: Muestra la información al usuario.
      3) Controller: Maneja las entradas del usuario.

2. Server (Master): Es el nodo que hace las veces de maestro en el clúster de Hadoop[18]. Es la máquina en la que se instala el servidor de BigTexts.
   a) Hadoop
      1) MapReduce: Componente de Hadoop para el procesamiento en paralelo.
      2) YARN: Componente de Hadoop para la planificación de tareas y gestión de recursos en clúster
      3) HDFS: Componente de Hadoop para la gestión distribuida de archivos.
   b) BigTexts: El servidor de BigTexts. Es el encargado de la gestión de la ejecución de las tareas de pre-procesamiento usando la infraestructura de Hadoop.
      1) BigTexts-UDFs: Son las implementaciones de las tareas de preprocesamiento, las cuales para poder ser usadas por Apache Pig, se tienen que programar con *User Defined Functions*[4]
   c) Pig: Apache Pig[19], provee un motor para la ejecución de flujos de datos en paralelo sobre Hadoop, hace uso directo de HDFS Y MapReduce [16].

3. Server (Slave): Representa cada una de las máquinas del clúster que hace las veces de esclavo, tiene instalado hadoop y sirve para el procesamiento en paralelo de las tareas.

[18]Para mayor información acerca de la configuración del clúster Hadoop, revisar el Anexo 7. Manual de instalación Hadoop

[19]Para mayor información acerca de la instalación de Apache Pig, revisar el Anexo 9. Manual de instalación Apache Pig

a) Hadoop: Es el sistema usado para el procesamiento en paralelo en la aplicación, el cual se compone de lo siguiente:

1) MapReduce: Es el estilo de computación usado para gestionar muchos cálculos de gran escala en una manera que es tolerante a fallos de hardware [36].
2) YARN: Es un *framework* para la planificación de tareas y gestión de recursos en clúster
3) HDFS: Es un sistema de ficheros distribuido que almacena los archivos a través de todos los nodos en un cluster Hadoop

5.4.5. Vista Física

Esta vista describe la correspondencia entre los componentes de Software y los de Hardware reflejando sus aspectos de distribución[22].

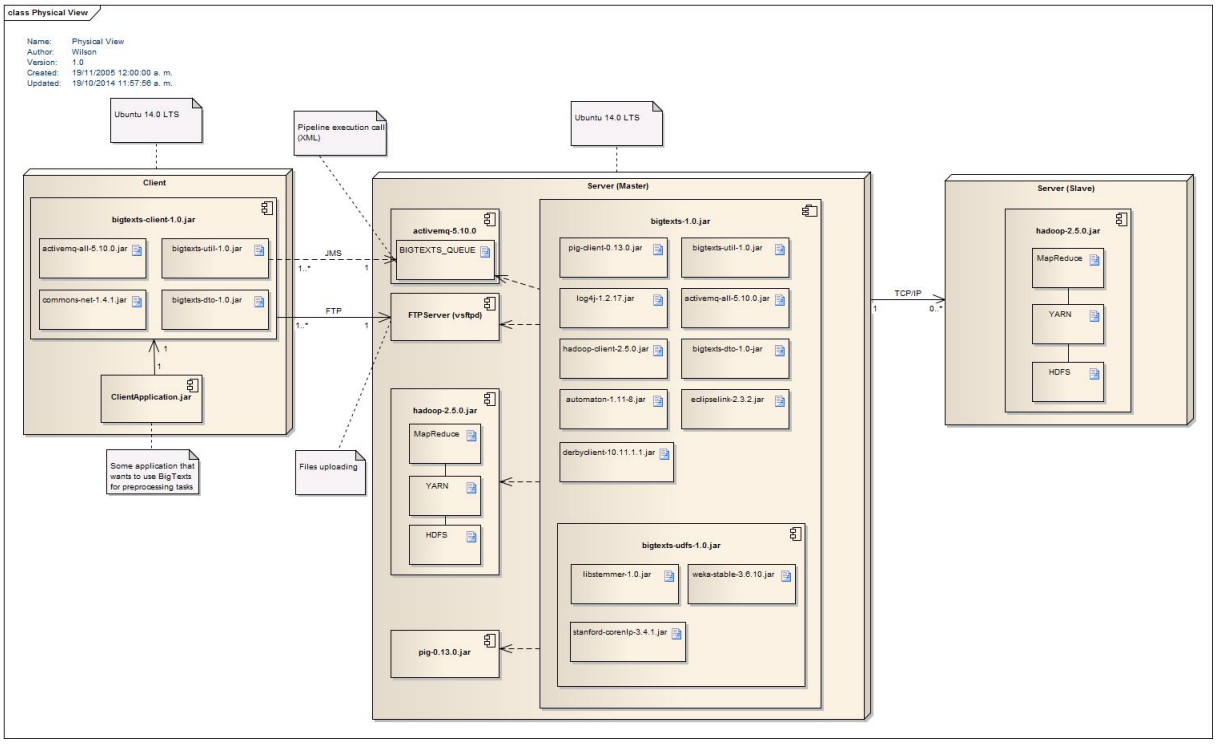


Figura 19: Vista física

1. Client: Máquina cliente en donde se encuentra instalada tanto la instalación de la aplicación externa, como el cliente de BigTexts.

   a) bigtexts-client.jar: Empaquetado con la aplicación cliente, la cual puede usarse por medio de una interfaz gráfica o como librería.

1) activemq-all-5.10.0.jar: Librería que permite el envío de mensajes al servidor de colas.
2) bigtexts-util-1.0.jar: Empaquetado con las clases de utilidades de BigTexts.
3) commons-net-1.4.1.jar: Librería para el envío de los documentos al servidor FTP.
4) bigtexts-dto-1.0.jar: Empaquetado con las clases que representan los objetos que comparten tanto el cliente como el servidor de BigTexts.

b) ClientApplication.jar: Representa la aplicación externa que usa el cliente de BigTexts como librería.

2. Server (Master): Máquina en la cual se instala la instancia principal de Hadoop, así como también el servidor de BigTexts.

a) activemq-5.10.0: Servidor de colas[20]
1) BIGTEXTS_QUEUE: Cola de mensajes mediante la cual se comunican los llamados desde el cliente hacia el servidor.

b) FTPServer (vsftpd)[21]: Es el servidor FTP que se instala en Ubuntu.

c) bigtexts-1.0.jar: Aplicación servidor, es la encargada de ejecutar las tareas de pre-procesamiento en la infraestructura BigData.
1) pig-client-0.13.0.jar: Es la librería que permite la conexión con la instalación de Pig.
2) bigtexts-util-1.0.jar: Empaquetado con las clases de utilidades de BigTexts.
3) log4j-1.2.17.jar: Gestor de la bitácora de la aplicación.
4) hadoop-client-2.5.0.jar: Librería que permite la interación con HDFS (Hadoop Distributed File System) y con Hadoop en general.
5) bigtexts-dto-1.0-jar: Empaquetado con las clases de comunicación entre el cliente y el servidor de BigTexts.
6) automaton-1.11-8.jar: Librería de autómatas de estado finito, usada por el cliente de Pig.
7) eclipselink-2.3.2.jar: Implementación de JPA (Java Persistence API) de Eclipse
8) derbyclient-10.11.1.1.jar: cliente del motor de base de datos liviano Apache Derby.

[20]Para mayor información acerca de la instalación del servidor de colas revisar el Anexo 5. Manual de instalación Active MQ

[21]Para mayor información acerca de la instalación del servidor FTP revisar el Anexo 6. Manual de instalación FTP Server

9) bigtexts-udfs-1.0.jar: Empaquetado con las implementaciones de las tareas de pre-procesamiento.
   a' libstemmer-1.0.jar: Librería para la implementación de Snowball Stemmer.
   b' stanford-corenlp-3.4.1.jar: Librería de Natural Language Processing usada para las implementaciones de las tareas de pre-procesamiento.
   c' weka-stable-3.6.10.jar: Librería para la implementación de Lovins Stemmer.

d) hadoop-2.5.0.jar: Instalación principal (maestra) del framework para procesamiento distribuido de conjuntos grandes de datos a través de clusters de computadores.
   1) MapReduce: Componente de Hadoop para procesamiento en paralelo.
   2) YARN: Componente de Hadoop para la planificación de tareas y gestión de recursos en clúster.
   3) HDFS: Componente de Hadoop para gestión distribuida de archivos.

e) pig-0.13.0.jar: Instalación de la librería para el acceso a Hadoop.

3. Server (Slave): Una de las máquinas secundarias (esclavas) en el clúster Hadoop.

4. hadoop-2.5.0.jar: Instalación principal (maestra) del framework para procesamiento distribuido de conjuntos grandes de datos a través de clusters de computadores.

   a) MapReduce: Componente de Hadoop para procesamiento en paralelo.

   b) YARN: Componente de Hadoop para la planificación de tareas y gestión de recursos en clúster.

   c) HDFS: Componente de Hadoop para gestión distribuida de archivos.

5.5. Construcción

El desarrollo de la aplicación se realizó en dos fases, una sin una metodología definida y otra con Personal eXtreme Programming (como se explicó en el capítulo 2.2). En el cuadro 7 se presenta el comparativo en la efectividad de la planeación, tanto en la fase sin metodología (Ad-hoc) como en la fase apoyada por PXP. El número de tareas sobre-estimadas, estimadas correctamente y sub-estimadas en la primera fase se desconocen, ya que no se realizaba estimación alguna. En ambas fases del desarrollo el tiempo real

fue mayor que el tiempo planeado, pero con la metodología PXP el porcentaje de desfase fue significativamente menor (11.67 % vs. 25.67 %)

	Desarrollo Ad-hoc	PXP
Número de tareas sobreestimadas	-	9
Número de tareas estimadas correctamente	-	31
Número de tareas subestimadas	-	10
Tiempo total planeado de desarrollo	60	48
Tiempo real de desarrollo	75.4	53.6
Diferencia entre el tiempo real y el planeado	15.4	5.6
Porcentaje de diferencia	25.67 %	11.67 %

Cuadro 7: Efectividad en la planeación

En el cuadro 8 se muestra un incremento en el tiempo dedicado a la planeación de las historias de usuario, así como también un incremento en el número de tareas técnicas, demostrando una definición de tiempos mucho más detallada, lo que en etapas posteriores permitió unos desfases menores y mayor calidad.

	Desarrollo Ad-hoc	PXP
Historias de usuario	30	43
Tareas técnicas	5	7
Tiempo total de planeación	26	37.1

Cuadro 8: Tiempos de planeación

En la tabla 9 se muestra una disminución significativa en el número de defectos incrementando la efectividad (Historias de usuario / Defectos) pasando de un porcentaje del 30 % a más del doble (69.76 %). Dado que en la primera fase de desarrollo no se crearon pruebas unitarias, el porcentaje de cobertura del código por pruebas unitarias (Unit Tests Code Coverage) fué del 0 %. En la segunda fase, al haber implementado Test Driven Development como parte de la metodología PXP, se incrementó significativamente el porcentaje de cobertura del código llegando al 75 %.

	Desarrollo Ad-hoc	PXP
Historias de usuario	30	43
Defectos	21	13
Efectividad	30 %	69.76 %
Code Coverage	0 %	75 %

Cuadro 9: Efectividad de desarrollo y code coverage

5.6. Aplicación

A continuación se presentan los pantallazos de la aplicación cliente, la cual se compone de cuatro pantallas, la primera (Figura 20) presenta la información general del proyecto y da la bienvenida al usuario.

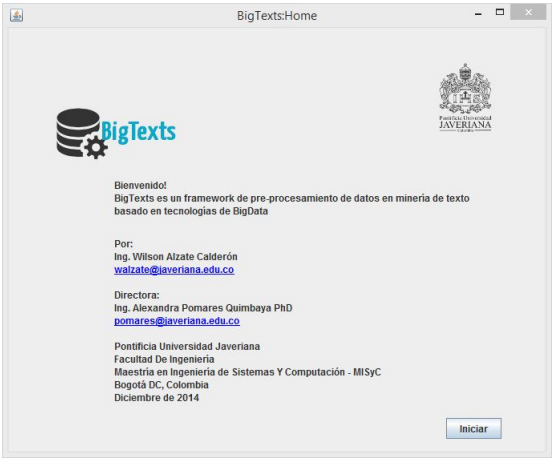


Figura 20: Pantalla de bienvenida

La segunda pantalla (Figura 21) permite la carga de los documentos que se quieren pre-procesar.

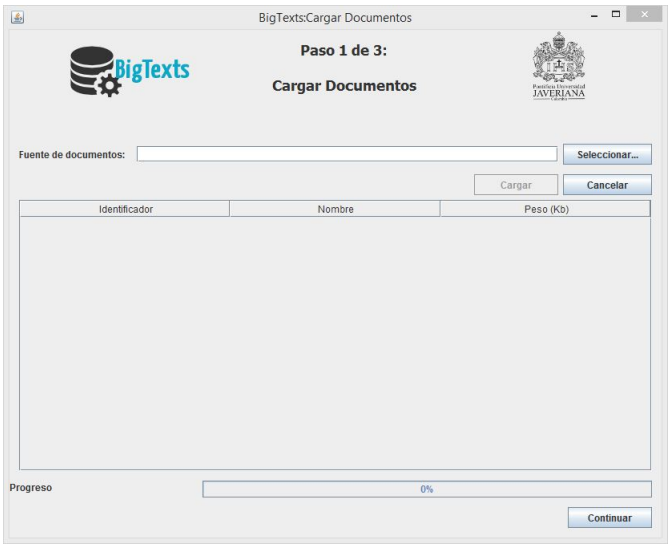


Figura 21: Paso 1 de 3: Cargar Documentos

La tercera pantalla (Figura 22) muestra el catálogo de tareas de pre-procesamiento disponibles, permite seleccionar un conjunto de ellas, esta-

blecer el órden de ejecución y parametrizarlas.

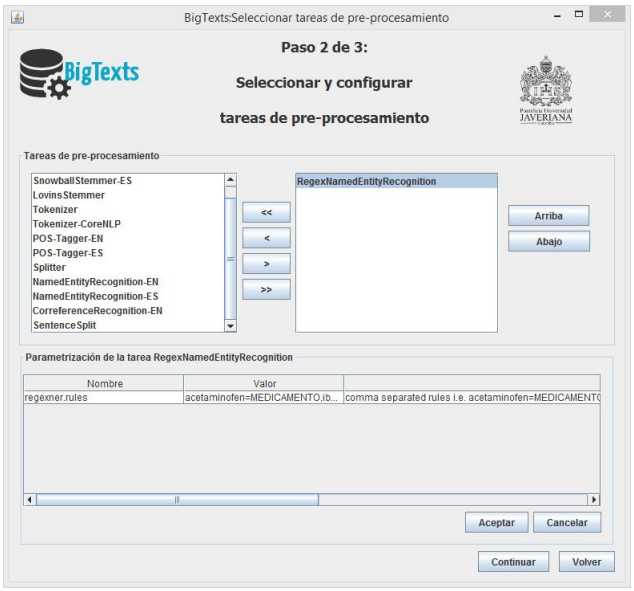


Figura 22: Paso 2 de 3: Seleccionar y Configurar Tareas de Pre-procesamiento

Y finalmente en la cuarta pantalla (Figura 23) se presenta el resumen de los documentos seleccionados, las tareas de pre-procesamiento escogidas, así como también su órden y parametrización en formato XML. Permite además la selección del destino (Si se desea que el resultado se entregue en el HDFS o en el FTP) y el botón de ejecutar, que envía la petición al servidor de BigTexts.

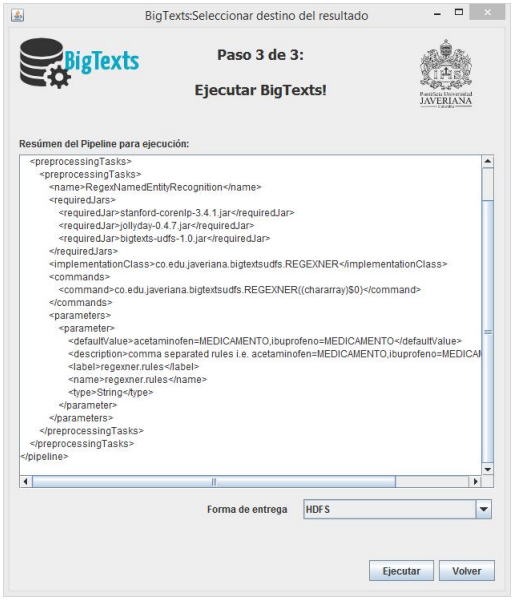


Figura 23: Paso 3 de 3: Ejecutar BigTexts!

6. VALIDACIÓN

A continuación se presenta la validación del sistema BigTexts para hacer cumplimiento del objetivo específico "Validar las funcionalidades provistas por el framework a través de su utilización en un caso de estudio aplicado al análisis de historias clínicas electrónicas del Hospital Universitario San Ignacio.".

Las máquinas provistas por el programa de maestría para realizar la configuración del clúster Hadoop (Cuadro 10), no son equipos con especificaciones altas (tipo servidor) sino más bien equipos de uso común. Se contó con una máquina principal (master) y cuatro máquinas secundarias (slaves). La aplicación cliente se instaló en la máquina master, así como el servidor BigTexts. Todas los computadores se instalaron con el sistema operativo Ubuntu 14.0.LTS. Las máquinas se conectaron en una red de área local (LAN) estableciendo direcciones IP estáticas. Se contó con 4 máquinas de marca HP y una Lenovo. La máquina master, que fué en donde se instaló el servidor de colas (ActiveMQ), el servidor FTP y las dos aplicaciones de BigTexts (el cliente y el servidor). El computador master contó con el doble de memoria RAM que el resto de equipos. Todas las máquinas contaban con procesadores Intel, 4 de ellas con Core 2 Duo y una con Core 2.

Nombre	Hostname	IP Interna	Marca	Memoria	Disco	Procesador
bigtexts-1	master	192.168.0.101	HP	3,8 GiB	155,3 GB	Intel Core 2 Duo CPU E7200 2.53GHz x 2
bigtexts-2	slave-2	192.168.0.102	HP	1.9 GiB	155.3 GB	Intel Core 2 Duo CPU E7200 2.53GHz x 2
bigtexts-3	slave-3	192.168.0.103	Lenovo	1.9 GiB	76.5 GB	Intel Core 2 CPU 4400 2.00GHz x 2
bigtexts-4	slave-4	192.168.0.104	HP	1.9 GiB	155.3 GB	Intel Core 2 Duo CPU E4600 2.40GHz x 2
bigtexts-5	slave-5	192.168.0.105	HP	1.9 GiB	155.3 GB	Intel Core 2 Duo CPU E7200 2.53GHz x 2

Cuadro 10: Máquinas del clúster

Se tomaron 4 archivos con registros de historias clínicas electrónicas del Hospital Universitario San Ignacio con los siguientes tamaños:

- 1.000 EHRs, 1.945.886 bytes (1.945 Mb)
- 700 EHRs, 1.423.803 bytes (1.423 Mb)

- 200 EHRs, 437.643 bytes (437.643 Kb) y
- 1 EHR, 761 bytes

Realizando 5 iteraciones para cada archivo en cada una de las siguientes tareas de pre-procesamiento:

- RegexNamedRecognition: Dada una lista de tokens con una etiqueta para cada una de ellas (medicamentos), el sistema identifica usando expresiones regulares si una palabra se encuentra en dicha lista y le establece la marca de MEDICAMENTO.
- Tokenizer: Partición de archivos en tokens.
- Tokenizer + POS-Tagger: Se realizan dos tareas de pre-procesamiento: Partición del archivo en tokens y luego (Part of Speech) identificando si cada uno de ellos es un verbo, un adjetivo, un adverbio, etc.
- Tokenizer + SnowballStemmer: Igualmente se realiza el particionado del archivo en tokens y luego se realiza la identificación de raices (Stemming) de cada uno de ellos.

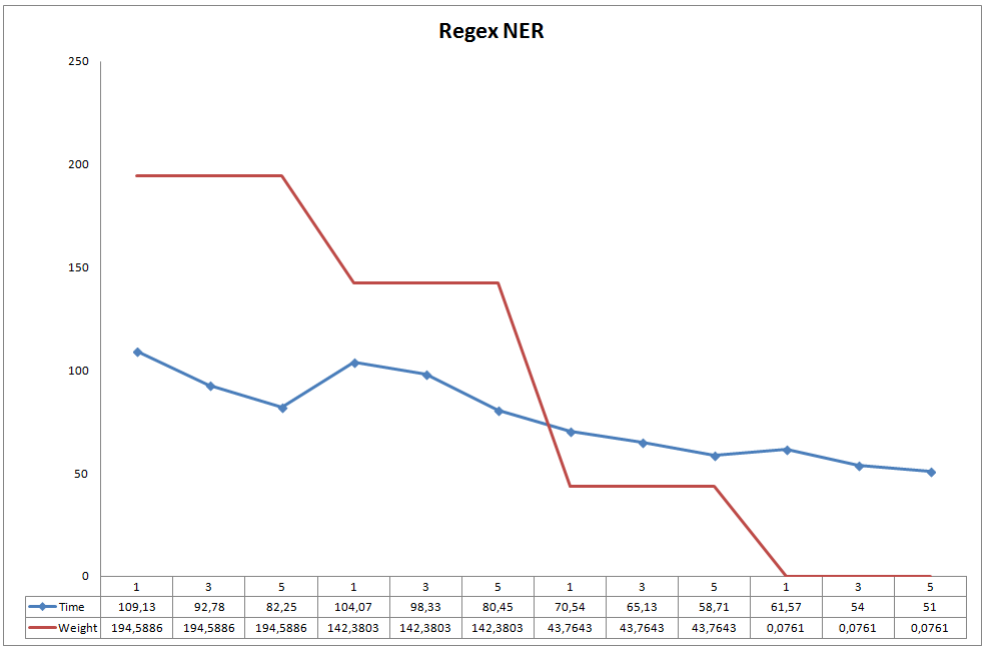
En el Cuadro 11 se muestran los promedios obtenidos de las ejecuciones, agrupadas por tarea de procesamiento, tamaño de archivo y número de máquinas disponibles para la ejecución en el clúster (Se realizaron activaciones y desactivaciones de máquinas en el cluster para tener escenarios de 1, 3 y 5 máquinas). Se decide realizar la comparación de esta manera para poder contar con la mayor similitud en los escenarios, usando las mismas características en las máquinas, las mismas librerías y por ende, los mismos algoritmos.

Preprocessing Task	Time (Seconds)	# Slaves	# Archives	Total Weight (bytes)
RegexNamedEntityRecognition	109,13	1	1	1.945.886
RegexNamedEntityRecognition	92,78	3	1	1.945.886
RegexNamedEntityRecognition	82,25	5	1	1.945.886
RegexNamedEntityRecognition	104,07	1	1	1.423.803
RegexNamedEntityRecognition	98,33	3	1	1.423.803
RegexNamedEntityRecognition	80,45	5	1	1.423.803
RegexNamedEntityRecognition	70,54	1	1	437.643
RegexNamedEntityRecognition	65,13	3	1	437.643
RegexNamedEntityRecognition	58,71	5	1	437.643
RegexNamedEntityRecognition	61,57	1	1	761
RegexNamedEntityRecognition	54	3	1	761
RegexNamedEntityRecognition	51	5	1	761
Tokenizer	50,22	1	1	1.945.886
Tokenizer	46,63	3	1	1.945.886
Tokenizer	36	5	1	1.945.886
Tokenizer	56,25	1	1	1.423.803
Tokenizer	41,33	3	1	1.423.803
Tokenizer	33,6	5	1	1.423.803
Tokenizer	46,67	1	1	437.643
Tokenizer	43,33	3	1	437.643
Tokenizer	39,4	5	1	437.643
Tokenizer	44,56	1	1	761
Tokenizer	40,2	3	1	761
Tokenizer	37,67	5	1	761
Tokenizer + POS-Tagger	181	1	1	1.945.886
Tokenizer + POS-Tagger	113,75	3	1	1.945.886
Tokenizer + POS-Tagger	85,67	5	1	1.945.886
Tokenizer + POS-Tagger	118,33	1	1	1.423.803
Tokenizer + POS-Tagger	96,67	3	1	1.423.803
Tokenizer + POS-Tagger	76,33	5	1	1.423.803
Tokenizer + POS-Tagger	80	1	1	437.643
Tokenizer + POS-Tagger	87	3	1	437.643
Tokenizer + POS-Tagger	54	5	1	437.643
Tokenizer + POS-Tagger	69,67	1	1	761
Tokenizer + POS-Tagger	65	3	1	761
Tokenizer + POS-Tagger	52,5	5	1	761
Tokenizer + SnowballStemmer	69,15	1	1	1.945.886
Tokenizer + SnowballStemmer	53,86	3	1	1.945.886
Tokenizer + SnowballStemmer	49	5	1	1.945.886
Tokenizer + SnowballStemmer	47,05	1	1	1.423.803
Tokenizer + SnowballStemmer	45,4	3	1	1.423.803
Tokenizer + SnowballStemmer	43,71	5	1	1.423.803
Tokenizer + SnowballStemmer	46,46	1	1	437.643
Tokenizer + SnowballStemmer	43,6	3	1	437.643
Tokenizer + SnowballStemmer	40,67	5	1	437.643
Tokenizer + SnowballStemmer	50,21	1	1	761
Tokenizer + SnowballStemmer	47,42	3	1	761
Tokenizer + SnowballStemmer	42,25	5	1	761

Cuadro 11: Datos de validación consolidados

A continuación se muestra el análisis de los datos obtenidos, usando figuras de líneas para identificar tendencias. Para la correcta visualización en la gráfica se dividió el tamaño del archivo por 10.000 y así tener dichos datos en la misma escala del tiempo.

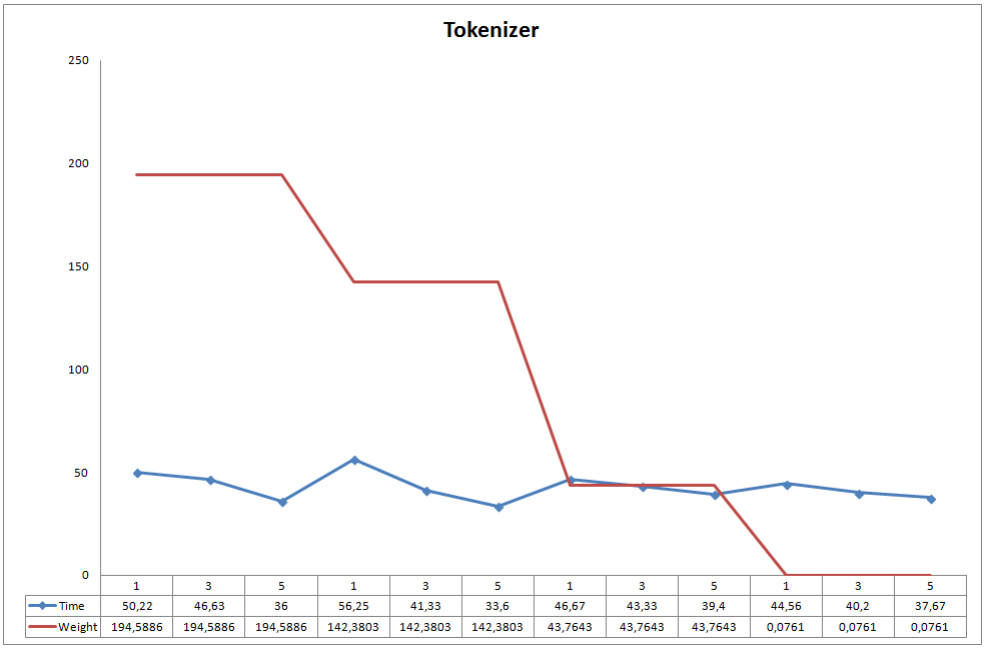
Encontrando por ejemplo para la tarea de Regex Entity Recognition (Figura 24) que con el archivo de 1.945.886 bytes que para una máquina el tiempo promedio para las iteraciones fue de 109,13 segundos, para tres máquinas de 92.78 segundos y para cinco de 82.25 segundos. Para el archivo de 1.423.803 bytes los tiempos fueron de 104.07, 98.33, 80.45 segundos. Así mismo, el archivo de 437.643 bytes tuvo tiempos de 70.54, 65.13, 58.71 y el de 761 bytes tiempos de 61.57, 54 y 51 segundos para una, tres y cinco máquinas activas en el clúster. Se muestra entonces una tendencia por archivo a la baja en tiempos al aumentar el número de máquinas disponibles en el cluster.



	1	3	5	1	3	5	1	3	5	1	3	5
Time	109,13	92,78	82,25	104,07	98,33	80,45	70,54	65,13	58,71	61,57	54	51
Weight	194,5886	194,5886	194,5886	142,3803	142,3803	142,3803	43,7643	43,7643	43,7643	0,0761	0,0761	0,0761

Figura 24: Validación - RegexNER

Para la tarea de Tokenización (Figura 25) para el archivo de 1.945.886 bytes se tuvieron tiempos de 50,22, 46.63, 36 segundos. Con el archivo de 1.423.803 los promedios de tiempos fueron de 56.25, 41.33 y 33.6 segundos. De la misma manera, para los archivos de 437.643 y 761 bytes se tuvo una mejora de tiempos de ejecución al aumentar el número de máquinas disponibles en el clúster.



	1	3	5	1	3	5	1	3	5	1	3	5
Time	50,22	46,63	36	56,25	41,33	33,6	46,67	43,33	39,4	44,56	40,2	37,67
Weight	194,5886	194,5886	194,5886	142,3803	142,3803	142,3803	43,7643	43,7643	43,7643	0,0761	0,0761	0,0761

Figura 25: Validación - Tokenizer

En cuanto a la tarea de Tokenización sumada a la de Part of Speech (Figura 26) para los archivos de 1.945.886 bytes (181, 113.75 y 85.67 segundos), 1.423.803 bytes (118.33, 96.67 y 76.33 segundos) y 761 bytes (69.67, 65 y 52.5) segundos se presentó mejoría en los tiempos al aumentar el número de máquinas disponibles en el clúster. Sin embargo, para el archivo de 437.643 bytes se presentó una desmejoría entre usar una y tres máquinas (tiempos de 80 y 87 segundos respectivamente) pero una mejora sustancial al usar finalmente las cinco máquinas con un tiempo de 54 segundos.

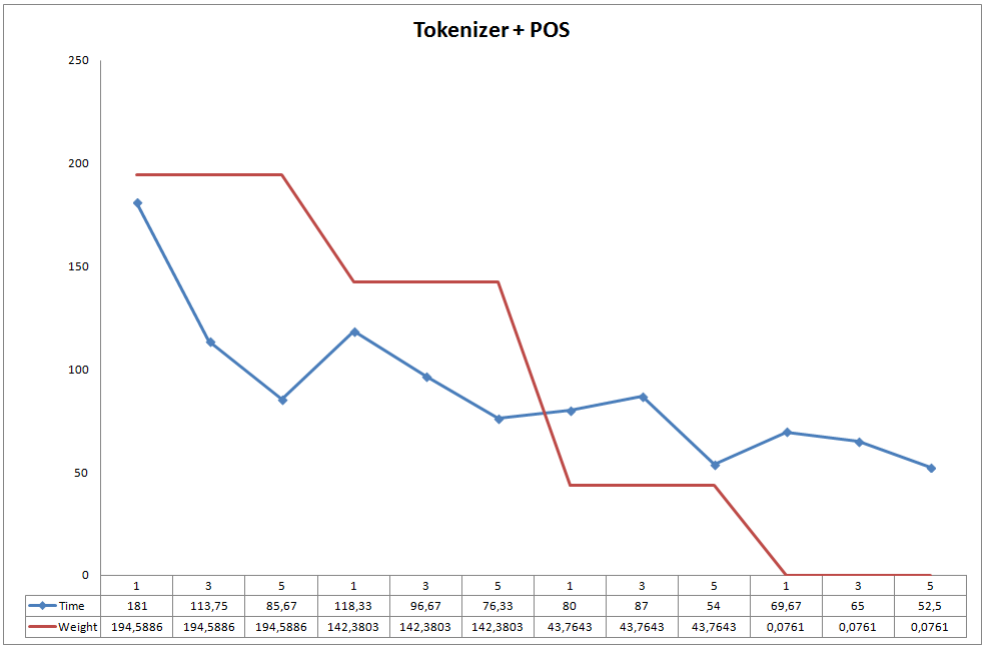


Figura 26: Validación - Tokenizer + POS

Para la tarea de Tokenización sumada a la de Stemming usando el algoritmo Snowball (Figura 27) se obtuvieron los siguientes tiempos para una, tres y cinco máquinas respectivamente:

- 1.945.886 bytes: 69.15, 53.86 y 49 segundos
- 1.423.803 bytes: 47.05, 45.4 y 43.71 segundos
- 437.643 bytes: 46.46, 43.6 y 40.67 segundos
- 761 bytes: 50.21, 47.42 y 42.25 segundos

Pudiéndose identificar una leve tendencia a la baja a la medida que se fueron adicionando máquinas disponibles al clúster.

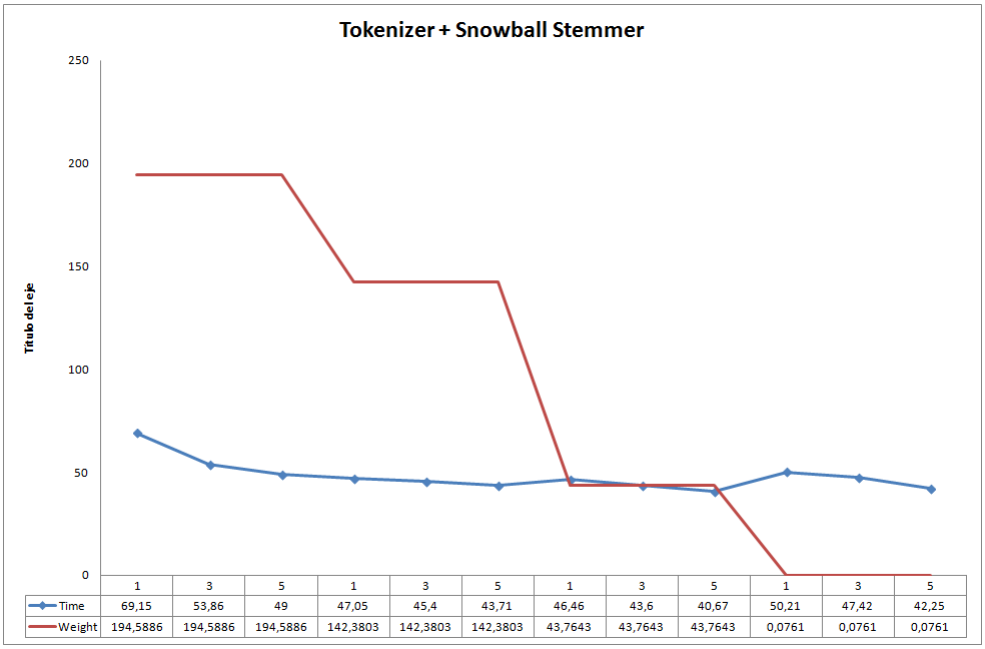


Figura 27: Validación - Tokenizer + Snowball

Como puede observarse los tiempos para un mismo archivo, en una misma tarea, difieren dependiendo del número de esclavos disponibles, mejorando a medida que se iban adicionando nodos esclavo. También se puede observar que la mejoría es mucho más notoria cuando se trabaja con tareas complejas como el RegexNER y Part Of Speech. Adicionalmente, se puede ver que entre más grande es el archivo, mayor es la velocidad de procesamiento, ya que por ejemplo para el de 761 bytes los tiempos para los diferentes escenarios (1,3 y 5 máquinas) es muy similar.

Se encuentra además que los resultados encontrados para cada uno de los escenarios (1, 3, 5 máquinas) fué el mismo, pudiendose aseverar con esto que el nivel de calidad en los resultados es confiable para cada uno de los escenarios.

Finalmente, siguiendo la metodología de la ciencia basada en el diseño, se procedió a realizar una validación de uso con un integrante del grupo de investigación Istar (Pontificia Universidad Javeriana) que trabaja en el proyecto Exemed; el ingeniero Julian Camilo Daza Rodriguez. El día 28 de noviembre de 2014, se le presentaron las funcionalidades del sistema BigTexts, especialmente la interfaz gráfica del cliente. El ingeniero realizó algunas observaciones que mejorarían la usabiliad del sistema, particularmente en la adición de listas desplegables para parametrización de las tareas de pre-procesamiento y en la adición de un mecanismo que permita comunicarle al cliente que la ejecución de las tareas de pre-procesamiento sobre los archivos seleccionados ha finalizado. En cuanto a la manera en la cual

se diseñó la interacción del cliente BigTexts con una aplicación externa, le pareció bastante sencilla la manera de creación del Pipeline usando el API de BigTexts. El uso de sólo tres pasos para la construcción del pipeline le pareció bastante sencillo y en general el sistema le pareció adecuado y útil, identificando que el mismo reemplazaría lo que actualmente realiza Gate en el sistema DISEArch, el cual usan para la búsqueda de registros médicos en historias clínicas electrónicas, con el fin de identificar cuáles de ellas pertenecen a un tipo específico de enfermedad en el Hospital Universitario San Ignacio.

Adicionalmente, como aporte a la base de conocimiento, actualmente se está escribiendo un artículo que espera ser sometido a una conferencia internacional y se creó un Blog (`bigtexts-misyc.blogspot.com`) en el cual se adicionaron algunas experiencias del proceso del presente proyecto de grado.

7. CONCLUSIONES

Con la realización del proyecto se pudo observar que la adición de una capa de abstracción a todo el ecosistema de tecnologías de Big Data y Minería de Texto es un aporte muy valioso, ya que la instalación, configuración e integración de los componentes es bastante dispendiosa. Esto se debe a que por ejemplo, las tecnologías de Big Data, aunque son poderosas, también son precarias en documentación precisa y confiable de instalación y configuración.

Por otra parte, el uso de la metodología PXP aunque incrementa significativamente el tiempo requerido para planeación y desarrollo de pruebas unitarias, mejora también la efectividad en el desarrollo, disminuyendo el número de defectos por artefacto (historias de usuario) ya que la adición de dichas pruebas, aunque es una tarea dispendiosa, permite garantizar la calidad del producto de software. Además, la estimación en las tareas, tanto técnicas (instalaciones, configuraciones, etc.) como de desarrollo, permiten tener un punto de partida muy valioso para próximas iteraciones, y por ende, una base mucho más sólida para las subsecuentes valoraciones de trabajo.

En cuanto a los resultados de la validación se pudo entender que vale la pena usar toda la infraestructura de Big Data siempre y cuando se tengan archivos lo suficientemente grandes o tareas lo suficientemente complejas, ya que se comprueba que para archivos grandes o tareas complejas, la adición de nodos en el cluster mejora notoriamente el rendimiento, pero que para archivos pequeños, por ejemplo, los tiempos son bastante similares a los obtenidos con una sola máquina.

Como trabajo futuro se propone usar Cloud Computing para abordar el problema planteado en el presente proyecto, con el fin de evaluar el contraste de esfuerzo de desarrollo, uso de nuevas tecnologías y escalabilidad de un enfoque como ese, comparado con el usado en BigTexts. Adicionalmente, otro campo de trabajo es el desarrollo de más UDFs para incrementar el catálogo de tareas de pre-procesamiento de BigTexts. Se propone también la instalación de un cluster Hadoop en la infraestructura del centro de computación de alto desempeño de la Pontificia Universidad Javeriana (ZINE)[22] y realizar pruebas con muchos más nodos.

[22] Al inicio del presente proyecto se realizó el contacto para usar dicha infraestructura, lo cual fué negado porque era un ambiente de producción.

ANEXOS

A continuación se presenta la lista de anexos del proyecto:

1. Entrevista de requerimientos
2. Documento de casos de uso
3. Estado del arte
4. Arquitectura BigTexts
5. Manual de instalación ActiveMQ
6. Manual de instalación FTP Server
7. Manual de instalación Hadoop
8. Manual de instalación PostgreSQL
9. Manual de instalación Apache Pig
10. Ejecución de pruebas funcionales
11. JavaDoc de la aplicación

Referencias

[1] Apache UIMA - apache UIMA. http://uima.apache.org/.

[2] Ravikant Agarwal and David Umphress. Extreme programming for a single person team. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, pages 82–87, New York, NY, USA, 2008. ACM. 00006.

[3] Apache Software Foundation. Hadoop. http://hadoop.apache.org/.

[4] Apache Software Foundation. User defined functions. http://pig.apache.org/docs/r0.13.0/udf.html.

[5] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley Professional, Boston, 1 edition edition, November 2002. 00016.

[6] Benjamin H. Brinkmann, Mark R. Bower, Keith A. Stengel, Gregory A. Worrell, and Matt Stead. Large-scale electrophysiology: Acquisition, compression, encryption, and storage of big data. *Journal of Neuroscience Methods*, 180(1):185–192, May 2009.

[7] Edward Capriolo, Dean Wampler, and Jason Rutherglen. *Programming Hive*. O'Reilly Media, 1 edition edition, September 2012.

[8] Nitesh V. Chawla and Darcy A. Davis. Bringing big data to personalized healthcare: A patient-centered framework. *Journal of General Internal Medicine*, 28(S3):660–665, June 2013.

[9] T.K. Das and P Mohan Kumar. Big data analytics: A framework for unstructured data analysis. *School of Information Technology and Engineering, VIT University*.

[10] Yani Dzhurov, Iva Krasteva, and Sylvia Ilieva. *Personal Extreme Programming – An Agile Process for Autonomous Developers*. Demetra EOOD, 2009. 00001.

[11] Ronen Feldman and James Sanger. *The Text Mining Handbook*. Cambridge University Press, 1 edition edition, March 2013.

[12] Bill Fox. Using big data for big impact. leveraging data and analytics provides the foundation for rethinking how to impact patient behavior. *Health management technology*, 32(11):16, November 2011. PMID: 22141243.

[13] Gartner. Platform as a service (PaaS). http://www.gartner.com/it-glossary/platform-as-a-service-paas.

[14] Gartner. Software as a service (SaaS). http://www.gartner.com/it-glossary/software-as-a-service-saas/.

[15] Gate. GATE. https://gate.ac.uk/.

[16] Alan Gates. *Programming Pig*. O'Reilly Media, 1 edition edition, September 2011.

[17] Claudia Gomez. Great: Modelo para la detección automática de relaciones semánticas entre resúmenes de texto provenientes de bases de datos del sector salud. http://pegasus.javeriana.edu.co/ PA123-03-SemantDatoSalud/anexos.html.

[18] AR Hevner, ST March, J Park, and S Ram. Design science in information systems research. *MIS QUARTERLY*, 28(1):75–105, March 2004.

[19] Hortonworks. HDP 2.0 - the complete hadoop 2.0 distribution for the enterprise. http://hortonworks.com/products/hdp/.

[20] IBM. Big data portfolio of products. http://www-01.ibm.com/software/data/bigdata/platform/product.html.

[21] KNIMEtech. KNIME text processing. http://tech.knime.org/knime-text-processing.

[22] P.B. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50, November 1995.

[23] Lancaster University. What is stemming? http://www.comp.lancs.ac.uk/computing/research/stemming/general/.

[24] Harshana Liyanage, Siaw-Teng Liaw, and Simon de Lusignan. Accelerating the development of an information ecosystem in health care, by stimulating the growth of safe intermediate processing of health information (IPHI). *Informatics in primary care*, 20(2):81–86, 2012. PMID: 23710772.

[25] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[26] Viktor Mayer-Schonberger and Kenneth Cukier. *Big data: a revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, Boston, 2013.

[27] Andrew McAfee and Erik Brynjolfsson. Big data: The management revolution. *Harvard business review*, 90(10):p60–68.

[28] William Q. Meeker and Yili Hong. Reliability meets big data: Opportunities and challenges. *Quality Engineering*, 26(1):102–116, January 2014.

[29] Microsoft. HDInsight. http://www.windowsazure.com/en-us/documentation/articles/hdinsight-get-started/.

[30] Keith D Moore, Katherine Eyestone, and Dean C Coddington. The big deal about big data. *Healthcare financial management: journal of the Healthcare Financial Management Association*, 67(8):60–66, 68, August 2013. PMID: 23957187.

[31] Travis B. Murdoch. The inevitable application of big data to health care. *JAMA*, 309(13):1351, April 2013.

[32] Oracle. Big data and oracle. http://www.oracle.com/us/technologies/big-data/index.html.

[33] Pivotal. Big data. http://www.gopivotal.com/big-data.

[34] Pivotal. Chorus. http://www.gopivotal.com/big-data/pivotal-chorus.

[35] Saptarshi Purkayastha and Jørn Braa. Big data analytics for developing countries – using the cloud for operational BI in health. *The Electronic Journal of Information Systems in Developing Countries*, 59(0), October 2013.

[36] Anand Rajaraman and Jeffrey D Ullman. *Mining of massive datasets*. Cambridge University Press, New York, N.Y.; Cambridge, 2012.

[37] Partho P. Sengupta. Intelligent platforms for disease assessment. *JACC: Cardiovascular Imaging*, 6(11):1206–1211, November 2013.

[38] SQA.net. ISO9126 - software quality characteristics. http://www.sqa.net/iso9126.html.

[39] V. Tablan, I. Roberts, H. Cunningham, and K. Bontcheva. GATECloud.net: a platform for large-scale, open-source text processing on the cloud. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983):20120071–20120071, December 2012.

[40] The Stanford NLP Group. Coreference resolution. http://nlp.stanford.edu/projects/coref.shtml.

[41] The Stanford NLP Group. Stanford log-linear part-of-speech tagger. http://nlp.stanford.edu/software/tagger.shtml.

[42] The Stanford NLP Group. Stanford named entity recognizer (NER). http://nlp.stanford.edu/software/CRF-NER.shtml.

[43] The University of Waikato. Weka 3 - data mining with open source machine learning software in java. http://www.cs.waikato.ac.nz/ml/weka/.

[44] Qiushi Wang and Zhao Yang. A method of selecting appropriate software architecture styles: Quality attributes and analytic hierarchy process. August 2012.

[45] WhatIs.com. What is data analytics (DA)? http://searchdatamanagement.techtarget.com/definition/data-analytics.

[46] WhatIs.com. What is NoSQL (not only SQL)? http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL.

[47] Paul Zikopoulos and Chris Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.* McGraw-Hill Osborne Media, 1 edition edition, October 2011.