Facultad 1

Sistema de clasificación de opiniones para medios de prensa digitales

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: *Duniel Reyes Castillo*

Tutores:

Ing. Tamara Betancourt Santana

Ing. Wendy Rodriguez Muñoz

Ing. Goar Espinosa Marrero

La Habana, Cuba

Junio 2017

Declaración de Autoría

Declaro ser el único autor del presente trabajo de diploma y confiero a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2017.

_____

Duniel Reyes Castillo

_____ _____

Ing. Wendy Rodriguez Muñoz Ing. Goar Espinosa Marrero

_____

Ing. Tamara Betancourt Santana

"No pretendamos que las cosas cambien, si siempre hacemos lo mismo.

La crisis, es la mejor bendición que puede sucederle a personas y países, porque la crisis trae progresos.

La creatividad nace de la angustia como el día nace de la noche oscura.

Es en la crisis que nace la inventiva, los descubrimientos y las grandes estrategias.

Quien supera la crisis se supera a sí mismo sin quedar superado.

Quien atribuye a la crisis sus fracasos y penurias, violenta su propio talento y respeta más a los problemas que a las soluciones. La verdadera crisis, es la crisis de la incompetencia.

El inconveniente de las personas y los países es la pereza para encontrar las salidas y soluciones. Sin crisis no hay desafíos, sin desafíos la vida es una rutina, una lenta agonía. Sin crisis no hay méritos.

Es en la crisis donde aflora lo mejor de cada uno, porque sin crisis todo viento es caricia. Hablar de crisis es promoverla, y callar en la crisis es exaltar el conformismo.

En vez de esto, trabajemos duro. Acabemos de una vez con la única crisis amenazadora, que es la tragedia de no querer luchar por superarla."

Albert Einstein.

1879

Dedicatoria

A mi padre José Reyes de Pino, estés donde estés papá te agradezco por todo el amor que me diste, por la educación que inculcaste en mi desde pequeño, por todos los abrazos y besos que me distes antes de partir hacia el cielo, por todo lo que fuistes y lo que inculcastes en mi para ser el hombre de bien que soy hoy día, a ti Papá, Gracias.

A mi madre Dora María Castillo, por ser una madre y un padre ejemplar para mi, por cuidarme y darme todo el amor del mundo, por la educación que me distes y por siempre apoyarme en mis desiciones, por levantarme cuando tropezaba, por consolarme cuando me sucedía algo, por ser mi sostén y mi apoyo, lo eres todo para mi, Te Amo Mamá.

Agradecimientos

A ti Papá, que estés donde estés el día de hoy ando seguro que te debes sentir el padre más orgulloso del mundo, gracias por el amor y el cariño que me distes cuando niño, por los regaños, los azotes cuando me los merecía, los besos en las mejillas, las caricias, por inculcarme tantas ganas de vivir y de amar, gracias por haber sido el modelo de padre que un hijo deseara ser, por siempre sonreir cuando las cosas iban mal y a pesar de faltarme hoy físicamente te llevo presente en mi corazón y siempre estarás en mi mente, a ti padre, Muchas Gracias por haberme dado la vida.

A ti Mamá, que fuiste todo para mi, fuiste madre cuando me tenías que cuidar para no llorar, padre para darme consejos de la vida y del amor, hermano para confiar plenamente en ti y contarte todos mis secretos, amigo para guiarme por el buen camino y sobre todo por ser la madre más cariñosa y especial del mundo. A pesar de no decirte te amo seguidamente hoy quiero darte el TE AMO más grande del mundo por haberme traído a la vida, por haberme criado con amor y cariño, por haberme educado correctamente como un hombre de bien y por todos los regaños y peleas que me dabas, una hoja no me bastaría para darte las gracias por todo lo que has hecho por mi, porque uno nunca podrá agradecer a una madre todo el amor y el cariño que deposita en sus hijos, así que hoy te doy las Gracias Mamá, gracias por ser mi madre.

A mis hermanos Lisbey, Jose Antonio, Yuniesky y Yaisel por haber sido un ejemplo a seguir por mi, por heberme ayudado en cualquier parte de mi vida y siempre escucharme cuando he tenido que hablar con ellos, por todo el amor y el cariño que nos tenemos y a pesar de las peleas siempre salir adelante en la vida, a ellos un beso y un abrazo.

A mi novia Nahomy, gracias nene por siempre haber estado a mi lado en los buenos y malos momentos, por siempre calmarme cuando llegaba a tu casa sin ganas de hacer nada y preocupado, gracias por pasarte las noches en vela a mi lado cada vez que tenía que hacer la tesis, gracias por tu apoyo y todo el amor que me has dado, gracias por haber soportado todos mis malos momentos y mis malas caras cada vez que algo me salía mal, gracias por darme tanto cariño y amarme de la manera que lo haces. TE AMO NENE.

A mis tíos Olga y Yoel y mis primas Saray y Saira por siempre darme ánimos y contar con ellos siempre que lo he necesitado.

A mis amigos Liomar, Yosvany, Niurka, Juan pablo, Luisbel, Tejera, Reiman, Anietsy , Anabel, Yilian, Rosaibis y Aldy por haber estado junto a mi estos años de la carrera, gracias por todo su apoyo y amistad, por haberme ayudado cada vez que lo necesitara y por siempre estar ahí a la hora que fuera necesario.

A mis compañeros de aula del 1504 por haber pasado momentos tan bellos a su lado, por los buenos y malos ratos que compartíamos juntos, por las fiestas, las locuras y todos los ratos agradables que de una manera u otra siempre estaban ahí.

A mis amigos de la facultad 1 y de todas las demás que de una manera u otra siempre me ayudaron a salir adelante.

A mis suegros Héctor y Yanitzia por todo el cariño y el amor que me han dado, por haberme acogido como un hijo más y darme ánimos cuando los he necesitado, por ayudarme durante toda la tesis de una forma u otra y por siempre estar ahí para mi. A mi cuñado David le agradezco por todo el apoyo moral y espiritual que me ha dado para salir adelante en este trabajo de diploma. A Keisy por ese maravilloso despertar húmedo que me da todas las mañanas y a Kiria por quererme tanto, a su manera ya que es muy resabiosa.

A la facultad 1 por haberme dado la posibilidad de hacer mi carrera aquí, es una de las mejores facultades de la Uci con un colectivo de profesores excelentes de los cuáles quiero agradecer a todos los que de una manera u otra estuvieron presentes durante mi trayectoria y me ayudaron de una manera u otra, especialmente a la decana Delly.

A mis tutores por toda la ayuda que me brindaron durante el proceso de realización de la tesis, por haberme aclarado las dudas cada vez que los iba a ver, gracias por siempre tener una sabia respuesta hacia mis dudas y haberme atendido a la hora que fuera necesario, Gracias.

Resumen

Los portales de noticias son uno de los principales medios de información que existe en la actualidad y cuentan con un gran volumen de contenido, entre ellos los comentarios que nos dan la posibilidad de conocer el estado de opinión de los lectores. En los últimos tiempos se han desarrollado sistemas internacionales y nacionales que permiten agilizar el proceso de clasificación de comentarios, a partir de un tema o contenido previamente especificado. Con el objetivo de dar solución a la clasificación de los comentarios y agilizar el proceso de conocimiento del estado de opinión de los usuarios, que actualmente se ejecuta de forma manual, se realiza un estudio y análisis de los Sistemas de Clasificación de Comentarios más empleados a nivel mundial atendiendo a los tipos de algoritmos que utilizan, clasificaciones, arquitectura y principales requisitos que poseen, para posteriormente definir los factores a tener en cuenta para la solución propuesta. Con el propósito de contribuir al desarrollo de la sociedad y los servicios al ciudadano, se desarrolla una herramienta de clasificación de comentarios que pretende automatizar el proceso de clasificación de comentarios en los medios de prensa digitales. Para el diseño y desarrollo de la propuesta de solución se seleccionaron como principales tecnologías: Python como lenguaje de programación, PyCharm como IDE de desarrollo, Tornado como servidor de aplicaciones, PosgreSQL como sistema gestor de Base de Datos y Visual Paradigm como herramienta para el modelado, guiado por la metodología AUP-UCI. La herramienta implementada posee un conjunto de características y funcionalidades que contribuyen a la clasificación de comentarios mediante una url, permitiendo lograr resultados más precisos y disminuir el tiempo su clasificación.

Palabra clave: agilizar, algoritmos, arquitectura, clasificación, comentarios, requisitos, Sistemas de clasificación de comentarios

Índice

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad se encuentra disponible un gran cúmulo de información a través de distintos medios electrónicos como bibliotecas digitales, colecciones de documentos y medios de prensa. La necesidad de acceder a esta información para su extracción y análisis, ha llevado a la creación de diversas formas de manipulación de información, entre las que se encuentra la clasificación de textos. Para que estos documentos digitales sean de fácil acceso se han tenido que organizar estas colecciones de tal forma que, permita su recuperación y análisis por medios automáticos. Sin embargo, el problema es complejo y se hace necesaria la continua investigación en la búsqueda de métodos y representaciones apropiadas para el proceso. Es por ello que se han surgido diversas líneas de investigación para el tratamiento automático de textos, entre las que se encuentran: la recuperación de información, la extracción de información, la búsqueda de respuestas y la clasificación de textos, entre otras (Saori, 2012).

La clasificación de textos es una tarea que facilita la organización de información y que consiste en determinar la categoría de un texto, entre varias categorías predefinidas, de acuerdo a ciertas características presentes en dicho texto. Inicialmente la clasificación de textos se realizaba de forma manual, sin embargo, realizar esta tarea de esta forma dificulta el proceso, eleba los costos y requiere del empleo de mayor tiempo. Por ello surge la necesidad de realizar la clasificación de estos documentos de forma automática. La mayor parte de los trabajos realizados para la Clasificación Automática de Textos se ha enfocado en clasificar textos por su tema, es decir, determinar a qué tema o temas pertenece un documento, entre varios temas dados (Cárdenas, Olivares y Alfaro, 2014).

En la última década ha surgido el interés de realizar la clasificación no-temática de textos, en donde lo que se busca es identificar el contexto y lo que expresan los usuarios de un tema determinado. Este trabajo de se enfoca concretamente en el problema de determinar la polaridad de opiniones, es decir, determinar que opiniones expresan algo a favor de un determinado tema, cuáles exprean oposición a lo que se plantea y cuáles representan una posición neutral, bajo un enfoque de Aprendizaje Automático utilizando características léxicas (Prieta, 2013).

Entre la clasificación de opiniones se encuentran tres subtareas principales. La primera de estas subtareas consiste en determinar si un texto dado contiene información objetiva o subjetiva. La segunda,

en determinar la orientación de una opinión[1] y la tercera y última subtarea en determinar la fuerza o grado de la opinión (Vilares Calvo, 2013). En particular, el presente trabajo se centra únicamente en determinar si un texto dado expresa una opinión a favor o en contra, o expresa neutralidad.

En la web podemos encontrar información de disímiles temas, ya sea de cultura, deporte, etc. hasta del acontecer diario de las naciones. Son precisamente los medios de prensa uno de los principales protagonistas en brindar dicha información y en donde podemos ejercer nuestros criterios. Teniendo en cuenta el creciente uso de la Internet y a la cantidad de usuarios que acceden a la misma, se hace muy difícil para los medios de prensa clasificar todos los comentarios que se generan en sus portales, constituyendo ello una tarea engorrosa que consume mucho tiempo.

El proceso de clasificación de comentarios actualmente en nuestro país se realiza de forma manual, trayendo consigo un alto costo y el empleo de mayor tiempo para realizar dicho proceso. Lo anterior provoca que los administradores de los portales de noticias tarden en conocer el estado de opinión de los lectores y les impida tomar una desición a tiempo sobre las conductas o informaciones que pueden cambiar en dependencia de la satisfacción del usuario con la noticia publicada. Es importante para los *webmaster* de estos sitios conocer a tiempo lo que opina un usuario sobre una noticia publicada y en dependencia de sus opiniones este puede poner al mercado un producto de primera necesidad con grandes éxitos, cambiar una medida que se tomó con un objetivo determinado, aumentar o no la cuota del mercado, saber que candidato tendrá más éxito en las elecciones, divulgar más información sobre un tema, conocer el cumplimiento de algo, abrir nuevos restaurantes en lugares determinados, entre otros. Al conocer lo que los usuarios comentan y sus necesidades se tienen en cuenta al realizar la toma de desiciones lo cuál trae consigo que los usuarios sientan una gran satisfacción al ver que sus sugerencias o comentarios son tomados en cuenta para cambiar, añadir o eliminar ciertas conductas, productos, informaciones, entre otros.

A partir de la situación problemática anteriormente planteada se define como **problema de investigación**: ¿Cómo realizar el proceso de clasificación de las opiniones en los medios de prensa para conocer el estado de opinión de los lectores?

[1] La orientación de una opinión es la forma de expresar si un texto dado corresponde con una opinión a favor o en contra(positiva o negativa respectivamente).

Por lo tanto, el **objeto de estudio** de la investigación se centra en el proceso de clasificación de opiniones. Delimitando así el **campo de acción** al proceso de clasificación de opiniones para medios de prensa digitales.

El **objetivo general** de este trabajo será: Desarrollar un sistema de clasificación de opiniones para los medios de prensa digitales.

Objetivos específicos:

1. Construir el marco teórico conceptual asociado al estudio del arte de los sistemas de clasificación de opiniones.
2. Diseñar el sistema de clasificación de opiniones para los medios de prensa.
3. Implementar un sistema que permita delimitar las fuentes de las opiniones relacionadas con la información indexada en los medios de prensa.
4. Validar el correcto funcionamiento de la propuesta de solución.

Para dar cumplimiento a los objetivos anteriormente planteados se definen las siguientes **tareas de la Investigación:**

1. Estudio de los conceptos y términos asociados al marco teórico de la investigación.
2. Caracterización de los algoritmos utilizados para la clasificación de opiniones.
3. Caracterización de los sistemas de clasificación de opiniones existentes.
4. Determinación de las fuentes a extraer de una opinión.
5. Determinación de las herramientas idóneas para extraer las fuentes seleccionadas en una opinión.
6. Determinación de la o las herramientas eficientes para determinar la orientación semántica (positiva, negativa, neutra) de una opinión.
7. Definición de la arquitectura del sistema.
8. Definición de los requisitos funcionales y no funcionales para el desarrollo del sistema de clasificación de comentarios.
9. Desarrollo de las funcionalidades que permitan identificar el tema de una noticia.
10. Desarrollo de las funcionalidades que permitan extraer las opiniones asociadas a una noticia.
11. Desarrollo de las funcionalidades que permitan extraer las fuentes de las opiniones.
12. Desarrollo de las funcionalidades que permitan identificar las opiniones por asunto.
13. Desarrollo de las funcionalidades que permitan almacenar la información referente de una opinión.
14. Validación del sistema desarrollado teniendo en cuenta su funcionamiento.

Luego de tratar los elementos fundamentales del área de la ciencia a incidir y los objetivos primordiales, se formula la siguiente **hipótesis de investigación:** El desarrollo del sistema de clasificación de opiniones para medios de prensa digitales permite reducir el tiempo que se demora en conocer el estado de opinión de los lectores con una alta precisión y logrando la escalabilidad del sistema. Teniendo en cuenta la hipótesis planteada se define como **variable independiente:** el sistema de clasificación de opiniones para medios de prensa digitales. Esta consiste en un servicio web que brinda la posibilidad de clasificar los comentarios que se encuentran en una dirección específica de la web, en positivos, negativos o neutros. Como **variables dependientes** se especifican: reducir el tiempo que se demora en conocer el estado de opinión de los lectores, dicha variable hace alusión al tiempo que demora un moderador de un sitio web en conocer el estado de opinión de los lectores en los medios de prensa; alta precisión, que expone que tan veraz es la clasificación efectuada por el sistema y la escalabilidad del sistema, que permita la integración o actualización de un módulo determinado sin afectar su funcionamiento.

Métodos teóricos:

1. **Inductivo-Deductivo:** Este método permitió llegar a conclusiones lógicas a partir de las observaciones de las soluciones existentes. Se utilizó para el planteamiento del objetivo general, la hipótesis de la investigación y la definición de algunos requisitos que son indispensables para comprender el sistema en su totalidad.
2. **Análisis Histórico-Lógico:** Este método permitió analizar los sistemas existentes relacionados con la moderación de los comentarios, determinando así las principales características, funcionalidades y atributos que deben ser tenidas en cuenta para el desarrollo de la propuesta de solución.
3. **Analítico-Sintético:** Este método permitió seleccionar los elementos más significativos relacionados con los sistemas de clasificación de opiniones, necesarios para el desarrollo del sistema en los medios de prensa.

Métodos empíricos:

1. **Entrevista:** Este método permitió definir las principales características y gran parte de los requisitos funcionales y no funcionales para el desarrollo del Sistema de clasificación de opiniones, además permitió recopilar información necesaria para el posterior desarrollo de la investigación.

2. **Observación:** Permite obtener una panorámica del problema existente y comprender las ventajas y desventajas de estos sistemas.
3. **Modelación:** Permite diseñar una base de datos que satisfaga el objetivo de la investigación y la realización del modelado de los diagramas necesarios para la implementación de la propuesta de solución.

El presente trabajo está estructurado en tres capítulos:

Capítulo 1: En este capítulo se plasman los elementos teóricos que soportan la investigación, se realiza el estudio de los sistemas homólogos que existen en el mundo, así como los principales métodos y herramientas que intervienen en el desarrollo del sistema.

Capítulo 2: En este capítulo se aborda la situación problemática y su propuesta de solución, se especifican los requerimientos funcionales y no funcionales y se describen a grandes rasgos los principales artefactos generados por la metodología de desarrollo de software seleccionada. Además se realiza la descripción de la arquitectura a utilizar y los patrones de diseño empleados en el desarrollo de la propuesta de solución.

Capítulo 3: En este capítulo se detallan los temas referentes a la implementación de la solución y se registran los resultados arrojados por las pruebas realizadas a la propuesta de solución con el objetivo de evaluar su correcto funcionamiento, así como, se hace la validación de la hipótesis trazada a inicios de la investigación.

Capítulo 1: Fundamentos teóricos del sistema de clasificación de opiniones para medios de prensa digitales

En el presente capítulo se abordan los aspectos relacionados con los sistemas de clasificación de opiniones, así como, los principales conceptos asociados al dominio del problema y que son necesarios para comprender cómo desarrollar el sistema. Además se realiza un estudio minucioso sobre las principales herramientas y tecnologías empleadas en el desarrollo de sistemas de este tipo.

1.1 Conceptos asociados al objeto de estudio

Clasificación de una opinión: Es la forma de clasificar una **opinión**, **parecer**, juicio o **consideración** que alguien hace acerca de otra persona o de algo en dependencia de algún tipo de clasificación determinada o mediante su polaridad. Dicha mención puede desarrollarse por vía oral o por escrito (Galarza, 2015).

Polaridad de una opinión: Desde el punto de vista de la minería de textos, el análisis de la polaridad de una opinión es una tarea de clasificación masiva de documentos de manera automática, en función de la connotación positiva o negativa del lenguaje ocupado en el documento. Es importante mencionar que estos tratamientos generalmente "se basan en relaciones estadísticas y de asociación, no en análisis lingüístico" (Astudillo, 2016).

Raíz de una palabra: La parte común de las palabras se llama raíz, también conocido como lexema. La raíz de una palabra es la parte de la palabra que no cambia. A partir de una raíz podemos formar palabras que son relacionadas por su significado (spanish, 2017). Un ejemplo de familia de palabras compuesto por la raíz boca sería: bocadillo, bocado, bocanada.

Palabras vacías o *Stop word*: Son todas aquellas palabras que carecen de un significado por si solas. Las palabras vacías suelen ser artículos, preposiciones, conjunciones, pronombres, etc. Es muy importante no asociar el concepto de palabras vacías como *"palabras malas o perjudiciales"*, ya que "son palabras necesarias que forman parte de nuestro lenguaje y facilitan la comunicación (Zeokat, 2014).

Base de conocimiento: Una base de conocimiento es un tipo especial de base de datos para la gestión del conocimiento. Provee los medios para la recolección, organización y recuperación computarizada de conocimiento. El más importante aspecto de una base de conocimiento es la calidad de la información que esta contiene. Las mejores bases de conocimiento tienen artículos cuidadosamente redactados que se mantiene al día, un excelente sistema de recuperación de información, y un delicado formato de contenido

y estructura de clasificación. Una base de conocimiento puede usar una ontología[2] para especificar su estructura(tipos de entidades y relaciones) y su esquema de clasificación. Una ontología, junto con un grupo de instancias de sus clases constituyen una base de conocimiento (Galarza, 2015).

Modelo neuronal: Es un tipo especial de red neuronal basado en un gran conjunto de unidades neuronales simples (neuronas artificiales), de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos. Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Cada unidad neuronal, de forma individual, opera empleando funciones de suma. Puede existir una función limitadora o umbral en cada conexión y en la propia unidad, de tal modo que la señal debe sobrepasar un límite antes de propagarse a otra neurona. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional (Gómez, 2014).

Colección de entrenamiento: Es un conjunto de documentos manualmente etiquetados con el sentido correcto de cada uno de los términos puede utilizarse para predecir el sentido de los términos que componen los nuevos documentos (Ureña, 2014).

1.2 Estudio de algoritmos y funcionalidades para realizar la clasificación

Un algoritmo es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema. De un modo más formal, un algoritmo es una secuencia finita de operaciones realizables, no ambiguas, cuya ejecución da una solución de un problema. Es el sistema por el cual se llega a una solución, teniendo en cuenta que debe de ser: definido, finito y preciso. Por preciso entendemos que cada paso a seguir tiene un orden; finito implica que tiene un determinado número de pasos, o sea que tiene un fin; y definido que si se sigue el mismo proceso más de un vez llegaremos al mismo resultado (utfsm, 2015).

[2] Una ontología es una definición formal de tipos, propiedades, y relaciones entre entidades que realmente o fundamentalmente existen para un dominio de discusión en particular.

1.2.1 Aprendizaje Automático

Cuando el ser humano adquiere conocimientos, habilidades, actitudes o valores a través del estudio, de la experiencia o la enseñanza, decimos que aprende. Este proceso es fácil para el humano, sin embargo, lograr que una máquina aprenda como lo hace el ser humano es una interrogante que existe desde los inicios de las computadoras. Actualmente no existe una máquina capaz de aprender de la misma manera que lo hace el hombre, sin embargo, se han creado algoritmos eficaces para algunas tareas de aprendizaje (DataCenter, 2017). En términos muy generales, podemos decir que un programa aprende si el desempeño obtenido para realizar alguna tarea, mejora con la experiencia. De manera formal. Se dice que un programa de computadora aprende de la experiencia E con respecto a una clase de tareas T y una medida de desempeño P, si su desempeño en las tareas T, medido con P, mejora con la experiencia E. Podemos decir entonces que el Aprendizaje computacional estudia los procesos computacionales que hay detrás del aprendizaje en humanos y en las máquinas. Esta disciplina juega un papel importante en muchas áreas de la ciencia (González, 2014).

1.2.2 Tipos de Aprendizaje

Un aspecto importante que influye en el aprendizaje es el grado de supervisión. En algunos casos, un experto en el dominio proporciona al aprendiz retroalimentación acerca de lo que es apropiado para su aprendizaje. En otros casos, a diferencia del aprendizaje supervisado, ésta retroalimentación está ausente, dando lugar al aprendizaje no-supervisado. Y en otros casos, se combinan el aprendizaje supervisado y el no supervisado, dando lugar al aprendizaje semi-supervisado (Gómez, 2014). A continuación se describe brevemente en que consiste cada tipo de aprendizaje.

1.2.2.1 Aprendizaje Supervisado

El aprendizaje supervisado es aquel en donde se intenta aprender de ejemplos como si estos fueran un maestro. Se asume que cada uno de estos ejemplos incluye características o atributos que especifican o definen a qué categoría o clase pertenece, de un conjunto de categorías o clases predefinidas, de esta manera cada ejemplo se asocia con su clase. Este tipo de aprendizaje es llamado supervisado por la presencia de los ejemplos para guiar el proceso de aprendizaje. Al conjunto de ejemplos del cual se intenta aprender se le llama conjunto de entrenamiento (Herrera, 2013).

Usando estos datos se construye un modelo de predicción, o aprendiz, el cual nos permitirá predecir la clase para nuevos objetos no vistos por el aprendiz. La construcción del modelo se logra gracias a un algoritmo de aprendizaje. Los algoritmos comúnmente utilizados son Naive Bayes, Máquinas de Vectores

Soporte (MVS), Vecinos más cercanos, J48 entre otros. En particular en el área de clasificación de textos, los algoritmos típicamente utilizados son Naive Bayes y Máquinas de Vectores Soporte (Herrera, 2013). Este tipo de aprendizaje tiene la ventaja de que no es necesario que al aprendiz se le muestren todos los ejemplos existentes, es decir que puede clasificar un ejemplo sin haberlo visto nunca. La desventaja es que a pesar de lo anterior, sí es necesaria una gran cantidad de ejemplos para el entrenamiento.

1.2.2.2 Aprendizaje no supervisado

Este tipo de aprendizaje no presupone ningún conocimiento previo sobre lo que se quiere aprender. Tampoco existe un maestro que conozca los conceptos a aprender, por esta razón a este tipo de aprendizaje se le denomina Aprendizaje no-supervisado. En el aprendizaje no-supervisado, los ejemplos sólo incluyen los atributos, es decir, no se encuentran asociados a una clase. Para este caso la tarea se enfoca en descubrir patrones comunes entre los datos, que permitan separar los ejemplos en clases o jerarquías de clases. De éstas se podrán extraer caracterizaciones, o permitirán predecir características, o deducir relaciones útiles, a lo que se denomina como agrupación (clustering). Algunos de los algoritmos más comunes son: Cobweb, EM, y Kmeans. Siendo este último el más utilizado en el área de Clasificación de Textos (Herrera, 2013). Este tipo de aprendizaje tiene la ventaja de que no es necesaria la presencia de un maestro para el aprendizaje o de un conjunto de entrenamiento.

1.2.2.3 Aprendizaje semi-supervisado

El aprendizaje semi-supervisado es la combinación del aprendizaje supervisado y el no-supervisado. En éste se aprende con la ayuda de dos conjuntos. Uno que contiene datos asociados a una clase, y el otro que contiene datos no asociados a una clase. La idea es aprender con los datos asociados a su clase y asociar una clase a los datos que no contienen asociada una clase. Algunos de los algoritmos más comunes son: *Co-training*, *ASSEMBLE* y *self-training* (Herrera, 2013).

1.2.2.4 Selección del tipo de aprendizaje

Se empleará en el periodo inicial un aprendizaje supervisado del sistema ya que el entrenamiento del mismo se realizará mediante ejemplos clasificados por expertos para guiar el proceso de aprendizaje y saber cómo se comporta el mismo en las diferentes etapas de la clasificación. Este aprendizaje consiste en hacer predicciones futuras basadas en comportamientos o características que se han visto en los datos ya almacenados (el histórico de datos). El aprendizaje supervisado permite buscar patrones en datos históricos relacionando todos los campos con un campo especial, llamado campo objetivo. Mediante su

uso podemos saber con exactitud cuál es la respuesta emitida por el sistema y si hubo existencia de errores en el proceso. Una vez analizado el comportamiento del sistema y la forma de clasificación se pretende utilizar el aprendizaje no supervisado en una segunda fase debido a que este se enfoca en descubrir los patrones que son comunes entre los datos existentes en la base de conocimiento y los que son entrados por el usuario a clasificar, permitiendo agrupar los datos en las clases existentes. De éstas se podrán extraer caracterizaciones, o permitirán predecir características, o deducir relaciones útiles, a lo que se denomina como agrupación (clustering).

1.2.3 Evaluación de los clasificadores

Una vez creado el o los clasificadores para alguna tarea, es importante conocer el desempeño de éstos, por lo que existen medidas de evaluación, entre las más comunes se encuentran la precisión, el recuerdo y la exactitud. La precisión es la probabilidad de que un documento etiquetado con la clase i corresponda realmente a esa clase. El recuerdo es la probabilidad de que un documento que pertenece a la clase i es etiquetado dentro de esa clase. La exactitud representa el porcentaje de las predicciones que son correctas (Esuli y Sebastiani, 2006).

Además, cuando se trata el aprendizaje supervisado es necesario tener un conjunto de datos independiente del conjunto de aprendizaje, para probar el clasificador, a este conjunto de datos se le llama conjunto de prueba. Sin embargo, muchas veces para evitar posibles sesgos en la selección de los conjuntos de entrenamiento y de prueba se utiliza la validación cruzada, que se explica a continuación (Esuli y Sebastiani, 2006).

1.2.3.1 Validación cruzada

La validación cruzada o *cross-validation* es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar cómo de preciso es un modelo que se llevará a cabo a la práctica (Minewiskan, 2017).

Este método es el más utilizado para estimar errores de predicción. Consiste en dividir el conjunto disponible en k subconjuntos, y se repite el proceso de entrenamiento y prueba k veces, cada vez utilizando una partición diferente de datos de entrenamiento y de prueba, y al final los resultados son promediados. Más específicamente los m ejemplos disponibles son divididos en k subconjuntos, cada uno de tamaño m/k.

La validación cruzada sólo produce resultados significativos si el conjunto de validación y prueba se han extraído de la misma población. En muchas aplicaciones de modelado predictivo, la estructura del sistema que está siendo estudiado evoluciona con el tiempo. Esto puede introducir diferencias sistemáticas entre los conjuntos de entrenamiento y validación. Si se lleva a cabo correctamente, y si el conjunto de validación y de conjunto de entrenamiento son de la misma población, la validación cruzada es casi imparcial (Minewiskan, 2017).

1.2.4 Extracción de las características del texto

La gran cantidad de textos existentes en formato el electrónico en los últimos tiempos, hace imposible que una persona pueda leer, comprender y sintetizar toda la información contenida en ellos. Para gestionar esta información, se aplican estrategias tales como la recuperación de información (RI) y la extracción de información (EI). En grandes conjuntos de textos, la RI intenta recuperar los textos relevantes según una consulta determinada, mientras que la EI encuentra y extrae información que satisfaga las necesidades de información de un usuario (Vallez, 2007).

La extracción de características generalmente consiste en tres etapas:

- Pre-procesamiento.
- Indexado.
- Reducción de dimensionalidad.

1.2.4.1 Etapa de Pre-procesamiento

El pre-procesamiento consiste fundamentalmente en eliminar aquellos elementos que generalmente no contienen información para la tarea de la clasificación. Consta de tres posibles fases básicas (Servertson, 2017):

- **Eliminación de etiquetas**. Si los documentos utilizados contienen algún tipo de etiquetas o cabeceras (ej. etiquetas de html o xml), estás podrán ser removidas, debido a que en algunos casos no proporcionan información útil para la clasificación.
- **Eliminación de palabras vacías**. Las palabras vacías son palabras que son muy frecuentes y que por lo general no contienen información, por ejemplo: pronombres, preposiciones, conjunciones, artículos, etc.
- **Lematización de palabras**. Por lematización nos referimos al proceso de remover los sufijos para reducir una palabra a su lema o raíz. Por ejemplo, comprender, comprenderlo y comprendió tienen la raíz comprend.

1.2.4.2 Etapa de Indexado

La representación de documentos más comúnmente empleada es la llamada modelo vectorial. En el modelo vectorial, los documentos son representados por vectores de palabras y una colección de documentos son representados por una matriz A (palabra por documento), donde cada entrada representa las ocurrencias de una palabra en un documento (Torres y García, 2016).

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}; \quad \text{matriz de ocurrencia de palabras en el documento}$$

Donde a_{ik} es el peso de la palabra i en el documento k.

Existen muchos caminos para determinar el peso a_{ik} de la palabra i en el documento k, pero muchas de las aproximaciones están basadas en dos observaciones empíricas:

- Entre más ocurre una palabra en un documento, más relevante es en el tema del documento.
- Entre más veces ocurre la palabra en los documentos de la colección, será menos relevante.

A continuación se describen los 3 esquemas de ponderado más comúnmente usados:

Ponderado Booleano (Castro, 2012)

Este es el esquema más simple y consiste en asignar 1 a a_{ik} si la palabra ocurre en el documento y 0 en otro caso: $a_{ik} = \begin{cases} 1 & si \;\; f_{ik} > 0 \\ 0 & en\ otro\ caso \end{cases}$

Donde f_{ik} es la frecuencia de la palabra i en el documento k.

Ponderado por frecuencia de palabras

Otro esquema simple es usar la frecuencia de la palabra en el documento: $a_{ik} = f_{ik}$

Donde f_{ik} es la frecuencia de la palabra i en el documento k.

Ponderado *TFxIDF* (Torres y García, 2016)

Los esquemas previos no toman en cuenta la frecuencia de la palabra en todos los documentos en la colección. Una aproximación bien conocida para calcular pesos de palabras es el TFxIDF (Term Frequency x Inverse Document Frequency), el cual asigna el peso a la palabra i en el documento k en proporción al número de ocurrencias de la palabra en el documento, y en proporción inversa al número de documentos en la colección en la que la palabra ocurre al menos una vez: $a_{ik} = f_{ik} \times \log\left(\frac{N}{n_i}\right)$

Donde f_{ik} es la frecuencia de la palabra i en el documento k, N el número de documentos en la colección y n_i el número de documentos en los que i aparece.

El ponderado booleano fue seleccionado para realizar el indexado en la etapa de extracción de las características del texto ya que el mismo nos posibilita validar la ocurrencia de las palabras en el texto asignándole 1 como valor y 0 en caso que la misma no exista.

1.2.4.3 Etapa de reducción de dimensionalidad

Un problema en la clasificación de textos es la alta dimensionalidad en el espacio de atributos, lo que hace que el procesamiento sea extremadamente costoso en términos computacionales. De ahí, que existe la necesidad de reducir el conjunto original de atributos, a este proceso se le llama reducción de dimensionalidad (Eikvil, 2016). Existen diversos métodos de reducción de dimensionalidad, a continuación se explican dos de ellos:

- **Filtrado**

Durante el filtrado de datos un subconjunto de datos es utilizado para representar un conjunto de datos más amplio y frecuentemente inmanejable. De forma similar a la selección de atributos, el filtrado de datos trata de eliminar información redundante para obtener buenos modelos con un volumen de datos manejable. Un caso diferente, aunque también puede considerarse como un caso de filtrado de datos, es cuando el investigador está interesado en un subconjunto de los mismos (AdSense 2014).

- **Lematización**

La lematización es un proceso lingüístico que consiste en, dada una forma flexionada (es decir, en plural, en femenino, conjugada, etc), hallar el lema[3] correspondiente. Es decir, el lema de una palabra es la palabra que nos encontraríamos como entrada en un diccionario tradicional: singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos (Benavides, 2014).

La lematización puede realizarse automáticamente mediante programas de análisis morfológico. Hay diversos grados de lematización posible: podemos hacer una lematización puramente morfológica, o bien hacer una lematización sintáctica que tenga en cuenta el contexto en el que aparece la palabra. Por ejemplo, en un análisis morfólogico la palabra ama tendría dos lemas: el sustantivo ama y el verbo amar.

[3] El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra.

Sin embargo, en un contexto sintáctico (es decir, en una oración), podemos desambiguarlo y optar por un único lema (Díaz, 2005).

Este proceso designa el análisis léxico de un texto con el fin de reagrupar las palabras reales de una misma familia. Por lo tanto, se reducen a una única entidad llamada «lema» o «forma canónica» las palabras de una misma familia. De este modo, la lematización consiste en reagrupar juntas las distintas formas de una palabra única. Esta reagrupa las distintas formas que puede adoptar una palabra: el plural, el verbo en infinitivo, el verbo conjugado en todos los tiempos, el nombre, etc.

1.2.5 Algoritmos de clasificación de opiniones

Existen una serie de dimensiones que se deben tener en cuenta para determinar cuál puede ser un algoritmo de clasificación razonable (Bravo y Rojas, 2015), entre ellas:

- Número de ejemplos en el conjunto de entrenamiento.
- Dimensionalidad del espacio de atributos o características.
- ¿Se supone que las clases son linealmente separable?
- ¿Los atributos son independientes?
- ¿Cabe esperar que los atributos dependan linealmente de la variable de destino?
- ¿Cuáles son los requerimientos de sistema en términos de velocidad / rendimiento / uso de memoria...?

Teniendo en cuenta las características anteriores y las problemáticas planteadas se decide estudiar los algoritmos más empleados en el tema de la minería de datos. Estos algoritmos permiten el manejar los datos en dependencia de las particularidades de cada uno. A continuación se describen las más importantes.

1.2.5.1 Naive Bayes

El clasificador probabilístico **Naive Bayes** (Acuna, 2015) o **Bayesiano ingenuo** es uno de los algoritmos de aprendizaje práctico más utilizados por su sencillez; entre sus ventajas está que su implementación es muy fácil y obtiene buenos resultados de clasificación en la mayoría de los casos .

Es una técnica de clasificación y predicción supervisada ya que necesita de ejemplos previos que nos ayuden a clasificar los datos a evaluar, Naïve Bayes nos permite construir modelos que predicen la probabilidad de resultados.

A pesar de tener una larga tradición en la comunidad de reconocimiento de patrones el clasificador Naive Bayes aparece por primera vez en la literatura del aprendizaje automático a finales de los ochenta con el objetivo de comparar su capacidad predictiva con la de métodos más sofisticados. De manera gradual los investigadores de esta comunidad de aprendizaje automático se han dado cuenta de su potencialidad y robustez en problemas de clasificación supervisada (Duda, Hart y Stork, 2001).

El clasificador Naive Bayes se construye usando el conjunto de entrenamiento para estimar la probabilidad de cada clase dados los valores de atributos (palabras) del documento de una nueva instancia. Para estimar las probabilidades se emplea el Teorema de Bayes: $P(c_j|d) = \frac{P(c_j)P(d|c_j)}{P(d)}$

El denominador en la ecuación anterior no distingue entre categorías y puede ser eliminado. Este método asume que los atributos son condicionalmente independientes, dada la clase. Esto simplifica los cálculos.

Fórmula del Teorema de Bayes para los atributos independientemente.

$$P(c_j|d) = P(c_j)\prod_{i=1}^{j} P(d_i|c_j)$$

Una estimación $\widehat{P}(c_j)$ para $P(c_j)$ puede ser calculada de la fracción de documentos de entrenamiento que es asignada a la clase c_j: $\widehat{P}(c_j) = \frac{N_j}{N}$

Donde N_j es el número de documentos de entrenamiento para los cuales la clase es c_j y *N* es el número total de documentos de entrenamiento.

Una estimación $\widehat{P}(d_i|c_j)$ para $P(d|c_j)$ está dada por: $\widehat{P}(d_i|c_j) = \frac{1+N_{ij}}{M+\sum_{k=1}^{M} N_{kj}}$

Donde N_{ij} es el número de veces de la palabra *i* ocurrida dentro de los documentos de la clase c_j en el conjunto de entrenamiento. Para evitar el problema de la probabilidad cero se utiliza Laplace[4] (agregar un 1). *M* es el número de términos en el vocabulario.

A pesar de que la suposición de independencia condicional es generalmente falsa para la aparición de la palabra en documentos, el clasificador Naive Bayes es sorprendentemente efectivo.

[4] La transformada de Laplace por definición es una integral que va desde menos infinito a más infinito.

1.2.5.2 Máquinas de Vectores Soporte (MVS)

Las máquinas de vectores soporte o máquinas de soporte vectorial (SVM, del inglés *Support Vector Machines*) (Suárez, 2014) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico y fueron introducidas en los años 90 por Vapnik y sus colaboradores [Boser et al., 1992, Cortes & Vapnik, 1995]. Aunque originariamente las SVMs fueron pensadas para resolver problemas de clasificación binaria, actualmente se utilizan para resolver otros tipos de problemas (regresión, agrupamiento, multiclasificación).

Dentro de la tarea de clasificación, las SVMs pertenecen a la categoría de los clasificadores lineales, puesto que inducen separadores lineales o hiperplanos, ya sea en el espacio original de los ejemplos de entrada, si éstos son separables o cuasi-separables (ruido), o en un espacio transformado (espacio de características), si los ejemplos no son separables linealmente en el espacio original. Como se verá más adelante, la búsqueda del hiperplano de separación en estos espacios transformados, normalmente de muy alta dimensión, se hará de forma implícita utilizando las denominadas funciones *kernel* (Suárez, 2014).

Las MVS han mostrado un buen desempeño en general en una gran variedad de problemas de clasificación, más recientemente en clasificación de textos. En términos geométricos, el problema que resuelve las MVS es identificar una frontera de decisión linear entre dos clases, a través de una línea que los separe, maximizando el espacio del hiperplano. Sin embargo, las MVS incluyen una función llamada kernel, la cual permite realizar separaciones no lineales de los datos, proyectando la información a un espacio de características de mayor dimensión. Esto se logra cambiando la representación de la función, mapeando el espacio de entradas *D* a un nuevo espacio de características $F = \{\emptyset(d) | d \in D\}$. Esto es:

$$d = \{d_1, d_2, \dots d_n\} \rightarrow \emptyset(d) = \{\emptyset(d)_1, \emptyset(d)_2 \dots, \emptyset(d)_n\}$$

En la figura 1 se muestra un mapeo de un espacio de entradas de dos dimensiones a un nuevo espacio de características de dos dimensiones, donde la información no puede ser separada por una máquina lineal mientras que en el nuevo espacio de características esto resulta sencillo.

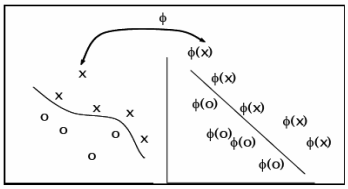


Figura 1. Mapeo del espacio de entradas a un espacio de mayor dimensión.

La función real ∅ no necesita ser conocida, es suficiente tener una función kernel *k*, la cual hace posible realizar el mapeo de la información de entrada al espacio de características de forma implícita y entrenar a la máquina lineal en dicho espacio. En términos geométricos un clasificador SVM binario (cuando se tienen solo dos categorías o clases), puede ser visto como un hiperplano en el espacio de características que separa los puntos que representan instancias positivas de la categoría de los puntos que representan instancias negativas. El hiperplano de clasificación se selecciona durante el entrenamiento como el único hiperplano que separa las instancias positivas conocidas de las instancias negativas conocidas con el máximo margen. El margen es la distancia desde el hiperplano al punto más cercano de los conjuntos positivo y negativo. Los hiperplanos de SVM son completamente determinados por un conjunto relativamente pequeño de instancias de entrenamiento conocidas como vectores de soporte.

1.2.5.3 Resultados del estudio de los algoritmos

Después de estudiar el funcionamiento de cada algoritmo se decide emplear el algoritmo de Maquinas de Soporte Vectorial como solución a la situación problemática, es un algoritmo que permite separar las clases previamente definidas mediante hiperplanos y realizar el entrenamiento del modelo mediante esta separación. Teniendo en cuenta que las noticias y los comentarios contienen cantidad de información, el algoritmo a diferencia del Naive Bayes nos permite procesar grandes volúmenes de datos tanto para el entrenamiento del modelo como para la clasificación del texto.

1.2.6 Herramientas de Procesamiento de Lenguaje Natural (PLN)

Se consideró un número de herramientas para el Procesamiento del Lenguaje Natural (Garcia, 2014), las cuales fueron evaluadas según las ventajas y desventajas que proporcionarían al desarrollo del proyecto teniendo en cuenta los objetivos que se deseaban alcanzar. Las herramientas utilizadas en este caso son la biblioteca NLTK (*Natural Language Tool-kit*) desarrollada en el lenguaje de programación Python y la librería Scikit-Learn, pero es necesario conocer las decisiones que llevaron a esta elección. A continuación se muestran las herramientas consideradas para el Procesamiento del Lenguaje Natural en la realización de este trabajo.

1.2.6.1 Stanford CoreNLP

Se trata de una herramienta de Procesamiento de Lenguaje Natural implementada en Java por la Universidad de Stanford (Finn, Kushmerick y Smyth, 2002). Fue una de las herramientas de mayor importancia a considerar, debido a la familiaridad que se tenía con la metodología de trabajo a la hora de

utilizarlo. Actualmente cuentan con modelos lingüísticos para el Chino, Inglés, Francés, Alemán y Español, el idioma objetivo de este trabajo. Una de las principales ventajas de esta herramienta es su implementación en Java, un lenguaje de más bajo nivel que Python o Ruby, lo que le proporciona una mayor capacidad de cómputo de datos.

1.2.6.2 Apache Lucene y Solr

Una de las grandes ventajas del sistema Solr es que posee una API (*Application Programming Interface*) flexible con la que se puede interactuar mediante los lenguajes de programación Ruby y Python, así como también es posible comunicarnos mediante el paso de mensajes JSON (JavaScript Object Notation), de manera similar a muchas arquitecturas servidor/cliente en la red (Diederich, 2012).

1.2.6.3 Apache OpenNLP

Esta herramienta utiliza una aproximación diferente a la que usa el Stanford CoreNLP. Se trata de un sistema desarrollado en Java, que permite ser utilizado como una biblioteca Java. Además de esto, posee una interfaz de programación en línea (scripting) que puede también usarse mediante la línea de comandos (Kessler, Numberg y Schütze, 2014). A pesar de encontrarse algo desfasada en comparación con el resto de opciones, permanece como una opción robusta y rápida de implementar.

1.2.6.4 GATE

El GATE (*General Architecture for Text Engineering*) es un sistema de software libre con una gran capacidad de procesamiento de texto. Fue desarrollado inicialmente por un equipo base de 16 programadores, empezando en 1995 como parte de un proyecto del EPSRC (*Engineering and Physical Sciences Research Council*), una organización basada en el Reino Unido con el objetivo de financiar la investigación científica en el país. Dado el largo tiempo que lleva esta herramienta en desarrollo, que abarca más de dos décadas, cuenta con un amplio abanico de funcionalidades de procesamiento de texto (Windsor, 2016).

1.2.6.5 Scikit-Learn

Scikit-learn es la principal librería que existe para trabajar con Aprendizaje Automático (Machine Learning), incluye la implementación de un gran número de algoritmos de aprendizaje. La podemos utilizar para clasificaciones, extracción de características, regresiones, agrupaciones, reducción de dimensiones, selección de modelos, o pre-procesamiento. Posee una API que es consistente en todos los modelos y se

integra muy bien con el resto de los paquetes científicos que ofrece Python. Esta librería también nos facilita las tareas de evaluación, diagnóstico y validaciones cruzadas ya que nos proporciona varios métodos de fábrica para poder realizar estas tareas en forma muy simple (Briega, 2015).

1.2.6.6 NLTK, *Natural Language Toolkit*

El NLTK (*Natural Language Toolkit*) es una biblioteca de Procesamiento de Lenguaje Natural que utiliza el lenguaje de programación Python. NLTK es software libre, lo que permite a estudiantes y al personal académico realizar estudios con la herramienta sin necesidad de realizar una inversión económica. Esta herramienta es también de código abierto, lo que lo hace ideal para expandir sus funcionalidades en caso de necesitarlo. El hecho de estar implementada como una biblioteca Python reduce la curva de aprendizaje, y la acerca al mundo académico, cuya mayor parte de integrantes se encuentra familiarizado con este lenguaje de programación (Hernández, 2016).

1.2.6.7 Herramientas de PLN elegidas

Una razón para la elección de NLTK como herramienta es el gran soporte que tiene, debido a las dimensiones de su comunidad de usuarios. Es una de las herramientas de Procesamiento de Lenguaje Natural de mayor aceptación en el ámbito científico. Otra razón importante es el apoyo que proporciona el libro *Natural Language Processing with Python*. Este libro tiene por autores a los creadores del NLTK, haciéndolo idóneo para comprender y utilizar todas las funcionalidades que aporta esta biblioteca. Se emplea para realizar el pre-procesamiento de las opiniones.

Además de la biblioteca NLTK también se utiliza la librería *Scikit-Learn* para realizar la creación y entrenamiento del modelo. Para ello se utilizan los algoritmos *CountVectorizer* que nos permite preparar los datos convirtiendo documentos de texto en una matriz de datos de recuento y *Pipeline* aplica secuencialmente una lista de transformaciones y un estimador final que permite crear el modelo a comparar con los comentarios. Para realizar la clasificación de los comentarios se utiliza el método *predict()* que busca la predicción de las ocurrencias de los comentarios en el modelo entrenado.

1.3 Análisis de soluciones homólogas

La clasificación de comentarios en internet hoy día se ha convertido en una necesidad para las personas encargadas de analizar toda la información que publican los usuarios en la red. A partir de los problemas y las necesidades existentes en nuestro país de tener un sistema que realice este proceso se decide realizar un estudio de sistemas homólogos que permita identificar los aspectos más significativos para la

creación de la nueva herramienta. A continuación se muestra el estudio realizado a los sistemas homólogos tanto en el ámbito internacional como nacional.

1.3.1 Sistemas de clasificación de comentarios a nivel internacional

BuscOpiniones

Es una herramienta desarrollada con el objetivo de poder recolectar, buscar y visualizar opiniones de una fuente elegida. Las opiniones son obtenidas de una base de datos generada a partir de artículos de prensa publicados en portales de noticias uruguayos (Dufort, Kremer y Mordecki, 2016).

Su funcionamiento básico como herramienta web ofrece la posibilidad de introducir un nombre (Mujica, Tabaré Vázquez, etc.) o identificador de una posible fuente (presidente, intendente, director técnico, etc.) y el tema del que se quiere recuperar opiniones (marihuana, transporte, etc.). Luego, mediante el envío de esos datos al servidor, se obtiene y visualiza en una línea de tiempo o las opiniones recolectadas por la plataforma de esa fuente y sobre ese tema. El sistema central consta de tres módulos principales: el que recolecta los artículos de la web y los indexa, el que permite realizar las búsquedas y desplegarlas en el navegador, y el que extrae las opiniones de cada artículo. La recolección se hace a través de un s*craper*[5] de noticias de las páginas web de los diarios El País, La República y El Observador. Las noticias luego son procesadas e indexadas en Solr (Dufort, Kremer y Mordecki, 2016).

Para permitir la búsqueda interactiva de opiniones, el sistema cuenta con un servidor web donde se encuentra instalada la aplicación, al cual se puede acceder desde un navegador. En la interfaz desplegada existen dos campos que permiten realizar la búsqueda, uno para la fuente y otro para el tema de la opinión, además de campos de búsqueda avanzada. Una vez ingresados los valores deseados, la web despliega en una línea de tiempo las opiniones extraídas por el módulo de extracción de opiniones de forma cronológica. El módulo de extracción de opiniones de los artículos es la columna vertebral del sistema. El trabajo de reconocer qué partes de los textos constituyen opiniones y la obtención de la fuente que emitió la opinión, fueron la motivación inicial del desarrollo de la plataforma, y donde se encuentra la aplicación del Procesamiento de Lenguaje Natural (Dufort, Kremer y Mordecki, 2016). La principal desventaja de este sistema es que es privativo y pese a la ventaja de poder acceder a su código fuente solamente se pueden acceder a las noticias que se encuentren publicadas en los diarios El país, La república y El observador ya que el scraper desarrollado solamente está definido para esos medios. El

[5] Un scraper es una técnica utilizada mediante programas de software para extraer información de sitios web.

sistema en su totalidad no es solución a la problemática anteriormente planteada ya que para poder realizar la clasificación es necesario entrarle al sistemas gran cantidad de datos.

Perspective

Perspective es una herramienta diseñada por Google y lanzada el 23 de febrero de 2017 que pretende identificar los comentarios tóxicos e, incluso, actuar en consecuencia. Todo mediante Inteligencia Artificial (Gimeno, 2017). La Inteligencia Artificial de Google localizada en Perspective no solo identifica los comentarios en base a los registros previos, también es capaz de utilizar el aprendizaje de máquina para ir añadiendo conocimiento a su base de actuación. De esta manera es capaz de utilizar ese mismo lenguaje calificado como tóxico para, por ejemplo, decirle en tiempo real a un comentarista que no le gusta su lenguaje.

Perspective puede, literalmente, utilizar ese mismo lenguaje que identifica y modera. La tecnología implementada aún está en sus primeras fases de desarrollo, pero ya la pueden utilizar tanto editores como desarrolladores. De hecho, Google la ha puesto en práctica con el medio *The New York Times* dándoles mayor rapidez a la hora de gestionar los comentarios y aumentando la fiabilidad contra el lenguaje tóxico (Antigua, 2017). Por el momento, la herramienta está disponible en inglés solamente, y es en ese idioma que realiza la clasificación de los textos. El proyecto prevé el desarrollo en la mayor cantidad de idiomas posibles en un plazo de no más de dos años en conjunto con otros desarrollos de Google.

1.3.2 Sistemas de clasificación de comentarios a nivel nacional

PosNeg Opinion

Es una herramienta nacional desarrollada en la Universidad Central de las Villas (UCLV) “Marta Abreu”. Este software clasifica la polaridad de la opinión, que consiste en determinar si la opinión es positiva o negativa con respecto a la entidad a la que se esté refiriendo. La aplicación PosNeg Opinion detecta de manera no supervisada la polaridad de opiniones siguiendo un esquema compuesto por cinco etapas: identificar tokens, desambiguar léxicamente cada token, obtener las acepciones de cada palabra, clasificar los tokens en positivo o negativo y evaluar la opinión. PosNeg Opinion detecta la polaridad de opiniones en español. El sistema desarrollado mostró un buen desempeño al clasificar la polaridad de 200 opiniones provenientes de foros de discusión en Yahoo.es, obteniéndose una exactitud de 0.965 y una precisión de 0.970 (Zeballos, 2013).

Permite que el usuario analice un gran cúmulo de opiniones de manera sencilla, convirtiendo los ficheros XML[6] a texto plano y se analizan independientemente las opiniones almacenadas en una lista. Para el usuario es transparente el procesamiento de los datos, así como la teoría y los algoritmos que se aplican en el análisis. Esta aplicación puede utilizarse como un módulo de una aplicación más general de minería de opinión, pues resuelve una de las fases de este proceso y es fácilmente reutilizable. Además, PosNeg Opinion se puede comunicar con otras aplicaciones mediante ficheros XML.

PosNeg Opinion fue desarrollada completamente en JAVA, por lo que es multiplataforma. Necesita como entrada un fichero XML con todas las opiniones a analizar y como salida muestra cuántas fueron positivas y cuántas negativas. A petición del usuario también retorna el porcentaje de las opiniones negativas y positivas así como una lista con las opiniones negativas y otra con las opiniones positivas. Adicionalmente, la aplicación destaca cuáles opiniones fueron las de mayor puntuación en cada caso (positivas/negativas) (Amores, Arco y Artiles, 2015).

Esquema general para la detección no supervisada de opiniones

La propuesta determina la polaridad de las oraciones teniendo en cuenta la polaridad de todas las acepciones de las palabras que se analizan al traducirse al idioma Inglés mediante un índice intralingüístico. Este esquema consta de cinco etapas como se muestra en la Figura 2.

Figura 2. Esquema general para detectar la polaridad de las opiniones.

El sistema pese a las características que posee es una herramienta que detecta de forma no supervisada la polaridad de las opiniones, pero solo es capaz de clasificar las opiniones expresadas en una oración o la opinión que expresa el texto en su totalidad. El análisis del sentimiento por tópicos es una combinación de dos áreas de investigación: la detección y seguimiento de tópicos con el área de la Minería de opinión y

[6] **XML** proviene de **eXtensible Markup Language** (**"Lenguaje de Marcas Extensible"**). Se trata de un **metalenguaje** (un **lenguaje** que se utiliza para decir algo acerca de otro) extensible de etiquetas que fue desarrollado por el **Word Wide Web Consortium** (**W3C**), una sociedad mercantil internacional que elabora recomendaciones para la World Wide Web.

Posneg Opinion no posee la funcionalidad de usar este tipo de análisis de sentimientos. El sistema no posee una herramienta avanzada para realizar la clasificación ya que no utiliza ningún algoritmo de agrupamiento de los indispensables en el tema de Minería de opinión lo que provoca una pérdida de precisión en la clasificación de los comentarios (Torres, Arco y Amores, 2015).

1.3.3 Análisis del estudio de los sistemas homólogos

Después de realizado el estudio algunos de los sistemas existentes en el mundo para la clasificación de los comentarios, este arrojó los siguientes resultados:

1. Los sistemas de clasificación estudiados no pueden ser utilizados para dar solución al problema planteado. Los sistemas nacionales analizados no cuentan con la documentación necesaria de cómo realizar el proceso de clasificación, además no describen los algoritmos y librerías que utilizan. En el caso de las herramientas internacionales no se puede utilizar su código fuente ya que son privativas.
2. Los sistemas de clasificación internacionales estudiados, presentan un filtro que permite descargar la información de las páginas web de manera más sencilla, sin embargo, de esta manera solamente se puede descargar la información de las páginas web de los diarios El País, La República y El Observador y no de otros sitios.
3. Los sistemas internacionales y nacionales no tienen disponible la base de conocimientos que emplean y no describen la forma de realizarla y utilizarla.
4. Debido a que estos sistemas no dan solución al problema planteado se evidencia la necesidad de contar con una herramienta que permita la clasificación de las opiniones en los medios de prensa nacionales y que brinde a los usuarios la posibilidad de guardar los datos clasificados.
5. El análisis del proceso de extracción de las características del texto de la herramienta nacional estudiada, permitió definir los pasos necesarios para realizar el proceso, como es el caso de la identificación de los *tokens* y la obtención de todas las acepciones de cada palabra del texto.

1.4 Tendencias y tecnologías de desarrollo

1.4.1 Lenguaje de desarrollo empleado

Los lenguajes de programación son herramientas que permiten crear programas y software. Es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras (Pérez, 2017). Durante el desarrollo del presente trabajo se analizaron los lenguajes de programación C++, Java, C# y Python de los cuales se trataron las características principales.

1.4.1.1 Lenguaje Visual C++

C++ es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje (Zator, 2015).

1.4.1.2 Lenguaje Java

Java está inspirado en C++ y se proyectó con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos ya sea a nivel de código fuente como a nivel de código binario. Es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana, genera ficheros de clases compiladas, pero estas son en realidad interpretadas por la máquina virtual de Java, quien mantiene el control sobre las clases que se estén ejecutando. El mismo es multiplataforma y seguro. Su sintaxis ha sido trabajada mejorando la de C++ logrando mayor sencillez y legibilidad. Presenta mayor robustez al simplificar la gestión de memoria y eliminar las complejidades del manejo explícito de punteros. Se puede compilar y ejecutar en cualquier plataforma de sistema operativo por ejemplo en Windows, Solaris o Linux gracias a su máquina virtual[7]. La ejecución de programas escritos en Java suele comportarse más lenta que la de aplicaciones de otro lenguaje haciendo un uso voraz de recursos como memoria y procesador. Esto se hace más notorio si la ejecución se basa en cálculos matemáticos complejos o si la aplicación presenta un diseño cargado de componentes visuales (Saula, 2014).

1.4.1.3 Lenguaje C#

Es un lenguaje de programación orientado al desarrollador y estandarizado por Microsoft como parte de su plataforma .NET[8], que más tarde fue aprobado como un estándar por la ISO. Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones (API), mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma (García, 2016). Algunas de las características que presenta son:

[7] Una **máquina virtual** es un software que emula un ordenador justo como si fuese uno real. Todo esto sucede en una ventana dentro de tu sistema operativo actual como cualquier otro programa que uses.

[8] .**NET** es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

- Sencillez y compatibilidad.
- Modernidad.
- Orientación a objetos y extensibilidad de operadores.
- Gestión automática de memoria.
- Instrucciones seguras.
- Extensibilidad de tipos básicos.

1.4.1.4 Lenguaje Python

Python, es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, por lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad (Álvarez, 2013).

En los últimos años el lenguaje se ha hecho muy popular, algunas razones son:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, ayudan a realizar muchas tareas habituales sin necesidad de programarlas desde cero.
- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.
- La cantidad de plataformas en las que es posible desarrollar, como Unix, Windows, OS/2, Mac, Amiga y otros.
- Es gratuito, incluso para propósitos empresariales.

1.4.1.5 Selección del lenguaje de programación más adecuado

El lenguaje de programación Python es seleccionado para la implementación del sistema siendo la razón principal que, cuenta con una extensa sección de librerías de gran utilidad para la realización del proyecto como NLTK para el procesamiento de lenguaje natural y Scikit-learn para el aprendizaje automático. Además, se valora su legibilidad, mantenibilidad y simpleza.

1.4.2 Entorno de desarrollo integrado (IDE)

Los entornos de desarrollo integrado (IDEs) son herramientas pensadas para escribir, compilar, depurar y ejecutar programas. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Hoy en día, los entornos de desarrollo

proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado como es el caso de C++, C#, Java, Python y Visual Basic (García, 2013).

Entre los entornos integrados de desarrollo que proporciona un marco de trabajo para el lenguaje seleccionado se decide utilizar PyCharm 2016.3. Este presenta un editor de códigos inteligente, que entiende los detalles específicos de Python y ofrece extraordinarios mejoradores de la productividad: formateo automático de código, finalización de código, refactorizaciones, importación automática, navegación de código con un solo clic, y mucho más. Además, es una herramienta multiplataforma y libre, es un IDE ligero y también presenta características de edición JavaScript[9], HTML y CSS[10].

1.4.2.1 PyCharm 2016.3

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web con varios precios que van desde los 29$ al año (andrearrs, 2014).

Algunas de las características principales son:

- Gestión de los intérpretes de Python con una nueva interfaz de usuario
- Nuevo soporte refinado para intérpretes remotos
- Soporte para Django 1.7
- Múltiples selecciones
- Nuevos *live templates* para Python
- Mejoras en soporte a *Vagrant*
- Nuevos *quick-fixes* automáticos
- Soporte completo de depuración en la consola interactiva Python
- Depuración para *Stackless* Python
- Soporte avanzado para AngularJS

[9] **JavaScript** (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

[10] ***CSS*** son las siglas de Cascading Style Sheets - Hojas de Estilo en Cascada - que es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación.

- IdeaVim actualizado con nuevas mejoras

1.4.3 Lenguaje de Modelado

Para el desarrollo del sistema se empleará como lenguaje de modelado el Lenguaje Unificado de Modelado UML , por sus siglas en inglés, es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (Orallo, 2015).
Mediante él se pueden especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos. Además puede conectarse con lenguajes de programación (ingeniería directa e inversa) y permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones) (Hernández, 2015).

1.4.4 Herramientas de Modelado

Entre las herramientas claves en el desarrollo o modelado de aplicaciones informáticas se encuentran las herramientas de Ingeniería de Software Asistida por Ordenador (CASE) (por sus siglas en inglés), las cuales son las encargadas de ayudar en el ciclo de desarrollo, con el fin de aumentar la productividad y reducir el costo en términos de tiempo y dinero (Mora, 2010).
Para el desarrollo de la propuesta de solución se utiliza Visual Paradigm en su versión 8.0 por sus características y la ventaja de ser software libre. Permite realizar ingeniería tanto directa como inversa y además es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como web o PDF[11] y es además una herramienta multiplataforma[12].

- **Visual Paradigm**

Es una de las herramientas CASE más utilizadas, es la suite creada por *Visual Paradigm International* (VPI). VPI es un proveedor de soluciones informáticas que incluye organizaciones para desarrollar aplicaciones de calidad, rápidas y baratas. Está compuesta por productos que facilitan a las organizaciones la visualización y diseño de diagramas. Sus soluciones se enfocan en eliminar la complejidad, aumentando así la productividad y disminuyendo el tiempo de desarrollo de las aplicaciones

[11] **PDF** (sigla del inglés Portable Document Format, «formato de documento portátil») es un formato de almacenamiento para documentos digitales independiente de plataformas de software o hardware. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto).
[12] Es soportada por varios sistemas operativos como Windows, Linux, etc.

informáticas. Visual Paradigm para UML (VP) en su versión 8.0 Enterprise Edition es una herramienta muy completa y permite la representación de varios artefactos (Fabian, 2014).

Algunas de las principales características de esta herramienta son:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, con diferentes especificaciones.
- Licencia: gratuita y comercial.
- Soporta aplicaciones Web y además se encuentra en varios idiomas.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS[13] y Subversión (control de versiones).
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XML.
- Ingeniería inversa - Código a modelo, código a diagrama.

1.4.5 Sistemas Gestores de Bases de Datos (SGBD)

Un SGBD, es una colección de programas cuyo objetivo es: servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos (Nipas, 2014). Las características de un Sistema Gestor de Base de Datos SGBD son (Nipas, 2014)

[13] Los archivos **CSV** (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla.

- **Abstracción de la información:** Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.
- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones.
- **Redundancia mínima:** Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- **Consistencia:** En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.
- **Integridad:** Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo y recuperación:** Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar, a partir de estas copias, los datos que se hayan podido perder.

1.4.5.1 MySQL

MySQL es un sistema de gestión de bases de datos relacional, bajo licencia GPL (*General Public License*) de la GNU[14]. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue

[14] **GNU** es un sistema operativo de tipo Unix desarrollado por y para el Proyecto **GNU**, y auspiciado por la Free Software Foundation. Está formado en su totalidad por software libre, mayoritariamente bajo términos de copyleft.

creada por la empresa sueca MySQL AB, que mantiene el *copyright* del código fuente del servidor SQL, así como también de la marca (Isocial web, 2014).

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración (Isocial web, 2014).

Características de MySQL

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's[15] en gran cantidad de lenguajes (C, C++, Java, PHP).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un buen nivel de seguridad en los datos.

Desventajas de MySQL (Sang, 2012)

MySQL surgió como una necesidad de un grupo de personas sobre un gestor de bases de datos rápido, por lo que sus desarrolladores fueron implementando únicamente lo que precisaban, intentando hacerlo funcionar de forma óptima. Por esta razón, aunque MySQL se incluye en el grupo de sistemas de bases de datos relacionales, carece de algunas de sus principales características:

- **Subconsultas:** tal vez esta sea una de las características que más se echan en falta, aunque gran parte de las veces que se necesitan, es posible reescribirlas de manera que no sean necesarias.
- **Triggers y Procedures:** Se tiene pensado incluir el uso de procedimientos almacenados en la base de datos, pero no el de triggers, ya que los triggers reducen de forma significativa el rendimiento de la base de datos, incluso en aquellas consultas que no los activan.

[15] La interfaz de programación de aplicaciones, abreviada como **API** del inglés: Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- **Transacciones:** a partir de las últimas versiones ya hay soporte para transacciones, aunque no por defecto (se ha de activar un modo especial).
- **Integridad referencial:** aunque admite la declaración de claves ajenas en la creación de tablas, internamente no las trata de forma diferente al resto de los campos.

1.4.5.2 PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (PostgreSQL, 2016).

Características principales de PostgreSQL

- Herencia de tablas y soporte para consultas con UNION, UNION ALL y EXCEPT.
- Herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL.
- Es una base de datos ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).
- Copias de seguridad en caliente (Online/hot backups).
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Soporte para distintos tipos de datos, como son: datos de tipo fecha, datos sobre redes (MAC, IP).
- Posee Interfaz de Programación de Aplicaciones (API's) para programar en C/C++, Java, .Net, Perl, Python, PHP.

1.4.5.3 Justificación del gestor de bases de datos seleccionado

Como SGBD se utilizará PostgreSQL, ya que es recomendado para sistemas que manejen gran cantidad de información. Sus características más importantes son: la estabilidad, potencia, robustez y facilidad de administración. Utiliza multiprocesos en vez de multihilos para garantizar la estabilidad del sistema, por lo que un fallo en uno de los procesos no afectará el resto y el sistema sigue funcionando. PostgreSQL está diseñado para entornos con altos volúmenes de tráfico/transacciones, presenta herramientas gráficas y de línea de comandos para diseñar bases de datos y administrarlas, requerimientos de administración y mantenimiento relativamente bajos con respecto el resto de bases de datos comerciales. Este gestor tiene un rendimiento excelente, es extensible y multiplataforma.

1.5 Metodología empleada para el desarrollo del sistema

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software, es un marco de trabajo empleado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Las metodologías se podrían clasificar en metodologías pesadas y metodologías ágiles. Las metodologías pesadas se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida. Las metodologías ágiles son métodos que permiten incorporar cambios con rapidez en el desarrollo de software y está orientada a un pequeño grupo de desarrollo con el mínimo de integrantes y trabajando en el mismo sitio. En muchas ocasiones, los modelos de gestión tradicionales no sirven para afrontar un reto que hoy en día resulta fundamental: incorporar cambios con rapidez y en cualquier fase del proyecto (Sommerville, 2011).

Por las características presentes en los tipos metodologías y la necesidad de documentar toda la información generada durante el desarrollo del sistema para su continuo desarrollo y perfeccionamiento por el centro CIDI de la Facultad 1 de la Universidad de las Ciencias Informáticas, el autor de la presente investigación decide emplear como guía para el proceso de desarrollo de la solución, la metodología de desarrollo ágil AUP-UCI por las características que esta presenta.

- **AUP-UCI**

La metodología de desarrollo de software AUP-UCI es una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Esta metodología surge como necesidad para los procesos productivos de la Universidad ya que al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) se hace necesario tener una guía de cómo realizar esta labor (Rodríguez, 2014).

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se apoyará en el Modelo CMMI-DEV v1.3 el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora; estas prácticas se centran en el desarrollo de productos y servicios de calidad (Rodríguez, 2014).

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre.

• **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

• **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

• **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.6 Herramientas de pruebas de software

Las pruebas de software (en inglés *software testing*) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada o *stakeholder*. Es una actividad más en el proceso de control de calidad (Campos, 2015).

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. Existen distintos modelos de desarrollo de software, así como modelos de pruebas. A cada uno corresponde un nivel distinto de involucramiento en las actividades de desarrollo (Plascencia, 2015).

1.6.1 Apache JMeter

JMeter es un proyecto de Apache que puede ser utilizado como una herramienta para realizar pruebas funcionales de carga, para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. JMeter es probablemente la herramienta más utilizada para realizar pruebas de rendimiento y *stress* sobre aplicaciones web, aunque también soporta otros protocolos como:

- Web – HTTP, HTTPS, SOAP, FTP, Database via JDBC, LDAP, Message-oriented middleware (MOM) via JMS, Mail-SMTP(S), POP3(S) and IMAP(S), MongoDB (NoSQL), Native commands or shell scripts, TCP.

JMeter está desarrollado con Java, por lo que puede utilizarse en todos los sistemas operativos con los que normalmente trabajamos (windows, mac o linux). Además de todo esto, comenzar a trabajar con JMeter es bastante sencillo. Lo que no es tan sencillo es saber leer los resultados que nos da la herramienta, por lo que en muchos casos, combinar una herramienta como JMeter con otra herramienta puramente funcional, nos puede dar más información que usar JMeter de forma aislada (Bruna, 2014).

Pruebas funcionales que se pueden realizar con JMeter

- Probar el rendimiento de recursos.

JMeter se puede utilizar para probar el rendimiento tanto de recursos estáticos como dinámicos: archivos, Servlets, scripts de Perl, objetos de Java, bases de datos y consultas.

- Construcción y ejecución de un plan de pruebas.

Se crea un plan de pruebas que será el encargado de describir un conjunto de pasos que ejecutará JMeter para realizar las diferentes pruebas.

1.6.2 Acunetix

Acunetix Web Vulnerability Scanner es una herramienta de seguridad de aplicaciones Web automatizada. Acunetix WVS es capaz de escanear cualquier sitio Web o aplicación Web que es accesible a través del protocolo HTTP / HTTPS. Sin embargo, no todas las pruebas se puede realizar de forma automática, y por lo tanto Acunetix WVS proporciona herramientas de Penetración manuales para pruebas particulares (Hackeruna, 2016).

- Acunetix es una herramienta automatizada de pruebas de seguridad de aplicaciones Web.
- Comprueba diferentes vulnerabilidades (por ejemplo inyección de SQL, *Cross Site Scripting*). Hasta la fecha Acunetix comprueba sobre más de 500 tipos diferentes de vulnerabilidades.
- Acunetix puede escanear cualquier sitio Web que es accesible a través del protocolo HTTP / HTTPS, básicamente, si el sitio Web se puede ver en un navegador, Acunetix puede escanearlo.
- Esta herramienta también proporciona herramientas de pruebas de penetración manuales que aumentan y contribuyen a las pruebas automatizadas, así como ayudar con la prueba de vulnerabilidades lógicas.

1.7 Conclusiones parciales

- Con el estudio y análisis realizado a los sistemas de identificación de opiniones en buscadores se fundamentaron las bases teóricas de la investigación, obteniendo los conocimientos sobre las características, así como, el correcto funcionamiento de estos sistemas.
- A partir de la investigación realizada se evidencia la necesidad de desarrollar un sistema que permita clasificar una opinión y caracterizarla en dependencia de su nivel, debido que los sistemas existentes a nivel internacional no brindan información sobre las tecnologías y herramientas que utilizan.
- El estudio realizado a diferentes metodologías de desarrollo de software nos permitieron seleccionar la metodología AUP-UCI por las características anteriormente planteadas, como herramientas se seleccionaron UML como lenguaje de modelado, Visual Paradigm 8.0 como herramienta de modelado, Python como lenguaje de programación, Pycharm 2016.3 como IDE para realizar el desarrollo, PostgreSQL como sistema gestor de base de datos y Acunetix y Apache Jmeter como herramientas de prueba de software.

Capítulo 2: Análisis y diseño del sistema de clasificación de opiniones para medios de prensa digitales

El objetivo de este capítulo es mostrar el diseño del sistema a desarrollar, se realiza un correcto análisis de todos los elementos que se relacionan con la clasificación de opiniones en los medios de prensa. Se desarrollarán las fases iniciales de la metodología de desarrollo AUP-UCI, inicio y planificación, además de presentar los diferentes artefactos, los cuales serán indicios cruciales para la entrega final del sistema, también se identifican los requisitos funcionales y no funcionales que deben estar presentes en la propuesta de solución.

2.1 Descripción de la propuesta de solución

El sistema está basado en un servicio web que permite clasificar las opiniones que se encuentran en los medios de prensa digitales. Una vez entrada la url y las etiquetas que el usuario desea seleccionar en un portal web específico, el sistema descarga toda la información proveniente de esas etiquetas mediante el uso de los selectores, permitiendo identificar un elemento dentro de una página Web para luego poder definir sus propiedades, abarcando desde el simple nombre de las etiquetas usadas en HTML (BODY, P, CODE, TABLE, UL, etc.) hasta complejas combinaciones que permiten un juego muy amplio de selecciones dentro de la página. Una vez descargada la información proveniente del sitio web se almacena en la Base de Datos y en un archivo XML para realizarle el pre-procesamiento al texto. El archivo XML posee una estructura para almacenar la información proveniente de esas páginas, como se muestra a continuación.

```
<?xml version="1.0" encoding="utf-8"?>
<noticia>
  <fecha>//fecha</fecha>
  <autor>//autor de la noticia</autor>
  <titulo>//titulo de la noticia</titulo>
  <sitio>//url del sitio</sitio>
  <texto>//Texto de la noticia</texto>
  <comentarios>
     <comentario>
       <fecha>//fecha del comentario</fecha>
       <autor>//autor</autor>
       <texto>
          //texto del comentario
       </texto>
    </comentario>
          .
          .
          .
  </comentarios>
</noticia>
```

Figura 3. Formato del archivo XML para almacenar la noticia y sus comentarios

Para realizar el pre-procesamiento es necesario que la noticia y los comentarios estén almacenados en la Base de Datos o en un archivo XML en la dirección local por ejemplo: /home/duniel/Escritorio/miner-opinion/data/xml. EL proceso está guiado por una serie de procedimientos que son necesarios ejecutar sin ninguna violación, de ocurrir alguna podría afectar el resultado final del sistema. Para este componente se utiliza la librería NLTK del procesamiento de lenguaje natural. Inicialmente se realiza la tokenización del texto o extracción de los signos de puntuación mediante el método *work_tokenize*, además se cambian todas las palabras a letra minúscula.

Una vez obtenida uniformidad en el texto se selecciona el lenguaje que presenta y se aplica el filtrado mediante el método *stopword,* que consiste en extraer del texto las palabras que no aportan información como los artículos, verbos, conjunciones, disyunciones, formas verbales, entre otras. Luego de realizado este proceso se obtiene una lista de palabras que si aportan valor al texto analizado, seguidamente se aplica el algoritmo *SnowballStemmer* en esta lista para obtener la raíz de las palabras y se obtiene un conjunto datos escritos en forma de diccionario {(*key*, *value*)}, donde *key* representa la variable o nombre que va a tener la colección y el atributo *value* tendrá una lista de estos lemas.

Una vez realizado el pre-procesamiento del texto se procede a entrenar el modelo con la librería *Scikit-Learn* y a crear el clasificador mediante el uso de las SVM. El método *CountVectorizer* prepara los datos convirtiendo el texto en una matriz de datos donde recibe los atributos que fueron devueltos en la etapa de procesamiento. La matriz se crea poniendo en cada fila el conjunto de datos correspondientes a las clases (positiva, negativa o neutro) definidas en el sistema para realizar el entrenamiento. Acto seguido se crea el clasificador utilizando el método *LinearSVC* que será el encargado de comparar los comentarios entrados por el usuario con el modelo. Obtenida la matriz y el clasificador se utiliza el método *Pipeline que* aplica secuencialmente una lista de transformaciones y un estimador final que permite crear el modelo a comparar, recibiendo la matriz y el clasificador como parámetro.

La llamada al método *pipeline.fit(samples, polarity)* crea el clasificador teniendo en cuenta que el atributo *samples* tendrá las muestras de las clases del conjunto de datos (positiva, negativa o neutro) y la *polarity* contiene números entre 0-2 que expresan que cuando una opinión es positiva es 0, cuando es negativa 1 y cuando es neutra 2. Obtenido el modelo entrenado y el clasificador se procede a realizar la clasificación de las opiniones que desee el usuario.

La clasificación está basada en determinar la polaridad (positivo, negativo o neutro) de las opiniones en una noticia determinada. Para realizar el proceso de forma exitosa es necesario que se ejecuten los

procedimientos vistos anteriormente, además de encontrar la información disponible en las fuentes requeridas. El proceso inicia buscando el archivo XML en la dirección especificada y los datos directamente de la BD, comprobando si estos existen. En caso de no existir lanza un error indicando que los datos no se guardaron correctamente, en caso contrario realiza el pre-procesamiento y entrena el modelo, generando así el clasificador.

El algoritmo *predict* es aplicado al clasificador para predecir el estilo de redacción de las opiniones y clasificarlas en dependencia de su orientación semántica. Cada vez que el clasificador encuentra una palabra la añade a una lista de opiniones (positiva, negativa o neutral) definidas en el sistema y calcula la probabilidad de ocurrencia del comentario en una escala entre 0-1, dando como clasificación las que obtuvieron mayor valor. La lista con los comentarios clasificados, (tanto positivos, negativos o neutros), se guardan para dar la respuesta mediante un archivo XML y se guarda la clasificación en la BD.

Al terminar el proceso de clasificación los datos son almacenados en la dirección física /home/duniel/Escritorio/miner_opinion/data/result mediante un archivo XML. El archivo tiene una estructura específica para devolver los resultados almacenando en cada una de las etiquetas los datos que se obtuvieron del proceso de clasificación. En la etiqueta <sitio> se almacena la url de la noticia y en la etiqueta <comentarios> se guarda una lista de todas las opiniones que fueron clasificadas como positivas, negativas o neutras. A continuación se muestra la estructura del fichero XML que dará la respuesta.

```
<noticia>
    <sitio>//url de la noticia </sitio>
  <comentarios>
    <positivos>
      <pos>//comentario positivo 1</pos>
        .
        .
    </positivos>
    <negativos>
      <neg>//comentario negativo1<neg>
        .
        .
    </negativos>
    <neutros>
      <neu>//comentario neutro1</neu>
        .
        .
    </neutros>
  </comentarios>
</noticia>
```

Figura 4. Formato del archivo XML para la respuesta de la clasificación

Además de la respuesta en el archivo XML el sistema muestra de manera visual todos los comentarios que fueron clasificados como positivos, negativos o neutros, además del autor de cada comentario.

Otro punto relevante de este trabajo es que se propone evaluar el método en el idioma español. Prácticamente, todos los trabajos previos que se han realizado dentro del área de clasificación de textos de opinión, se ha evaluado en textos en el idioma inglés. Por supuesto, esto conlleva la necesidad de conformar un *corpus*[16] para el español. Actualmente no existe un *corpus* etiquetado para el idioma español, indispensable para utilizarse en el proceso de Aprendizaje Automático (Castillo, Cardenas y Curti 2017). Como una consecuencia de lo anterior, en este trabajo se dan los primeros pasos para la creación de este *corpus*, definiendo una serie de criterios para el etiquetado del *corpus*. Dado el costo de la creación de este corpus el presente trabajo también exploró la posibilidad de un enfoque de aprendizaje supervisado, disminuyendo la necesidad de un gran *corpus* de entrenamiento etiquetado.

2.2 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en el que trabaja el sistema. El modelo de dominio se describe mediante diagramas UML (especialmente mediante diagramas de clases). De manera general el modelo del dominio ayuda a los usuarios, clientes, desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se sitúa el sistema (Larman, 2004).

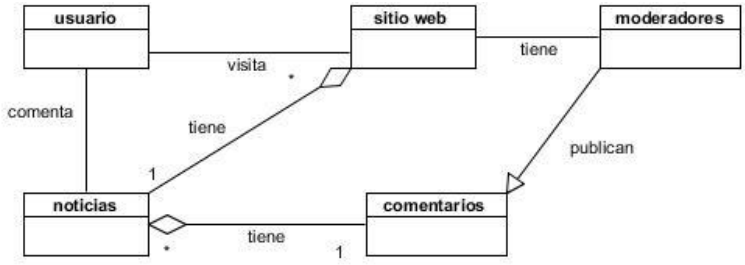


Figura 5. Modelo de dominio

Descripción de los elementos del modelo de dominio

Usuario: Son los usuarios que interactúan con los portales web en busca de información.

[16] Un corpus es un conjunto de textos relativamente grande, creado independientemente de sus posibles formas o usos. Es empleado en la Inteligencia Artificial para el entrenamiento y clasificación de textos.

Sitio web: Es la plataforma donde los usuarios pueden obtener variedades de información y emitir un criterio sobre algo.

Noticias: Son aquellas que informan a los usuarios sobre un tema determinado.

Comentarios: Son las opiniones emitidas por los usuarios sobre un tema determinado.

Moderadores: Son los encargados de analizar los comentarios de los usuarios y realizar la publicación de estos en los portales web.

2.3 Levantamiento de requisitos

La obtención de los requisitos es un paso muy importante para las siguientes etapas del desarrollo del software, pues un error en estas fases iniciales puede ocasionar que el sistema no cumpla las expectativas de los usuarios y difícilmente aporte valor agregado al negocio para el que debe ser concebido.

Los proyectos de software son altamente vulnerables y se afectan de manera crítica cuando alguna de estas actividades se desarrolla de manera pobre lo que conduce a darle una mayor importancia a los requerimientos dentro del proceso completo. Fallar en la descripción o en el entendimiento de un requerimiento, puede llevar a gasto extra de esfuerzos y de tiempo (Amaya, 2013).

2.3.1 Requisitos funcionales del sistema

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Pressman y Troya 2010a).

En el proceso de levantamiento de requisitos (Tabla 1) para el desarrollo del Sistema de clasificación de opiniones para los medios de prensa digitales fueron detectados un total de 7 requisitos funcionales, 3 de prioridad muy alta para el cliente, 1 de prioridad alta, 2 de prioridad media y 1 de prioridad baja.

Tabla 1. Listado de Requisitos Funcionales (RF)

Listado de Requisitos Funcionales (RF)	
RF1 Extraer noticias.	**RF2.3 Entrenar el modelo.**
RF2 Clasificar comentarios.	**RF2.4 Clasificar comentarios.**
RF2.1 Pre-procesar texto.	**RF3** Visualizar comentarios.
RF2.2 Crear el modelo.	

Tabla 2. Resumen de la etapa de captura de los RF

Prioridad para el cliente	Cantidad de requisitos funcionales
Muy Alta	3
Alta	1
Media	2
Baja	1
Total	7

2.3.2 Requisitos no funcionales del sistema

Los requerimientos no funcionales del sistema son aquellos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema (Pressman y Troya, 2010).

Los requerimientos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware o a factores externos como los reglamentos de seguridad, las políticas de privacidad, entre otros.

- **Software**

RNF1: El sistema debe ser multiplataforma.

RNF2: El sistema debe tener instalado el servidor web Apache 2 y el servidor de aplicaciones Tornado.

- **Hardware**

RNF3: El servidor de aplicaciones web y de base de datos deben poseer como mínimo un CPU Core i3 de 4ta generación a 2.20 GHz con 16 Gb de RAM DDR3.

RNF4: El servidor de aplicaciones debe poseer una capacidad mínima de 30 GB para el almacenamiento de los datos.

RNF5: El servidor de bases de datos debe tener una capacidad mínima de 15 GB**.**

- **Restricciones de diseño**

RNF6: El lenguaje de programación a usar será Python.

RNF7: Como herramientas de desarrollo se emplea el PyCharm 2016.3.

RNF8: Como herramienta para el modelado de UML se utiliza Visual Paradigm 8.0.

- **Mantenimiento y Soporte**

RNF9: Las operaciones de mantenimiento del sistema no deberán exceder de 1 día.

RNF10: El sistema debe ser escalable, permitiendo incorporarle nuevas funcionalidades sin afectar las existentes.

- **Usabilidad**

RNF11: La plataforma podrá ser empleada por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

- **Confiabilidad**

RNF12: En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad del mismo.

RNF13: Para garantizar la fiabilidad del sistema, se debe contar con un mecanismo de salvas externas para la información que maneja el mismo.

- **Seguridad**

RNF14: Protección contra acciones no autorizadas o que puedan afectar la seguridad de los datos.

RNF15: Protección del código fuente mediante la restricción del acceso a este creando una interfáz de logueo con un usuario y una contraseña.

2.4 Historias de usuarios (HU)

Las HU son utilizadas en AUP-UCI para especificar los requisitos del software desde el punto de vista del cliente, es decir, estas historias son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. Las HU deben tener el detalle mínimo para que el programador pueda realizar una estimación poco riesgosa del tiempo que llevará el desarrollo del software (pmoinformática, 2013). A continuación se muestra la descripción de las historias de usuarios más importantes.

Tabla 3. Historia de usuario Cargar Noticia

Historia de usuario	
Número: *1*	**Nombre Historia de Usuario:** *Cargar noticia*.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Duniel Reyes Castillo*	**Iteración asignada:** *1*
Prioridad en negocio: *Muy Alta*	**Puntos estimados:** *2/5*

Riesgo en desarrollo: *Medio*	**Puntos reales:** *2/5*
Descripción: *El sistema permitirá cargar las noticias que se encuentran publicadas en los medios de prensa.*	
Observaciones: *Para poder cargar las noticias al sistema es necesario que la máquina se encuentre conectada a la red y con acceso a los medios de prensa nacionales.*	

Tabla 4. Historia de usuario Clasificar comentarios

Historia de usuario	
Número: *2*	**Nombre Historia de Usuario:** *Clasificar comentarios*.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Duniel Reyes Castillo*	**Iteración asignada**: *1*
Prioridad en negocio: *Muy Alta*	**Puntos estimados:** *3/5*
Riesgo en desarrollo: *Medio*	**Puntos reales:** *3/5*
Descripción: *El sistema clasificará la orientación semántica (positiva, negativa, neutra) de cada comentario.*	
Observaciones: *Cuando el sistema realiza la clasificación de una opinión (positiva, negativa, neutra) la añade a una lista definida por el sistema, cuando el proceso termina de analizar todas las opiniones el sistema da la respuesta mediante un fichero XML que almacena en la dirección física /home/duniel/Escritorio/miner_opinion/data/result* y además a travéz de la interfáz principal del sistema.	

2.5 Modelo de Datos

Un modelo es un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia. Es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de integridad, las cuales definen un conjunto de estados consistentes. El modelo de datos puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos, además para la manipulación de consultas y datos, así mismo es el elemento clave en el diseño de la arquitectura de un manejador de la base de datos (BD) (Pérez, 2013). A continuación se muestra el modelo de datos de la propuesta de solución:

Figura 6: Modelo de datos.

A continuación se describen cada una de las tablas creadas en el modelo de datos:

Tabla *webpage*: Esta tabla hace referencia a los selectores que son necesarios almacenar en el sistema para realizar la extracción de la información de los diversos medios de prensa. Está compuesta por los atributos *domain* para extraer el dominio de la noticia, *author* para extraer el autor, *date* para la fecha de publicación del comentario, *morenewselector* para acceder a los comentarios en las demás páginas referentes a la noticia y el *attrmorenews* para extraer los comentarios de esas páginas, *contentselector* para extraer el contenido de la noticia y finalmente *commentselector* para extraer los comentarios.

Tabla noticia: La tabla noticia representa las noticias que serán añadidas al sistema para procesar, compuesta por un identificador sitio, título y texto.

Tabla comentario: Esta tabla representa los comentarios que se añadirán de cada noticia una vez que se almacenen en la BD, compuesta por un identificador id, autor, fecha, polaridad (positiva, negativa o neutra), texto y noticiasitio que hace referencia a la url donde se encuentra publicada la noticia.

2.6 Descripción de la arquitectura de software y los patrones de diseño

La arquitectura de software es una vista del sistema que incluye los principales componentes del mismo, alcanzando a través de los mismos la visión del sistema. Establece además, los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades (Bernardo, 2010).

2.6.1 Arquitectura basada en componentes

El desarrollo de software basado en componentes permite reutilizar piezas de código pre elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión. La reutilización de software es un proceso de la Ingeniería de Software que conlleva al uso recurrente de activos de software en la

especificación, análisis, diseño, implementación y pruebas de una aplicación o sistema de software (Matriam, 2014).

Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Características de un componente

- **Identificable:** Debe tener una identificación que permita acceder fácilmente a sus servicios que permita su clasificación.
- **Auto contenido:** Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- **Puede ser remplazado por otro componente:** Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- **Con acceso solamente a través de su interfaz:** Debe asegurar que estas no cambiaran a lo largo de su implementación.
- **Sus servicios no varían:** Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.
- **Bien Documentado:** Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.
- **Es genérico:** Sus servicios deben servir para varias aplicaciones.
- **Reutilizado dinámicamente:** Puede ser cargado en tiempo de ejecución en una aplicación.
- **Independiente de la plataforma:** Hardware, Software, S.O.

Las tecnologías de objetos proporcionan el marco de trabajo técnico para un modelo de proceso basado en componentes para la ingeniería del software. El paradigma orientado a objetos enfatiza la creación de clases que encapsulan tanto los datos como los algoritmos que se utilizan para manejar los datos. Si se diseñan y se implementan adecuadamente, las clases orientadas a objetos son reutilizables por las diferentes aplicaciones y arquitecturas de sistemas basados en computadora (Matriam, 2014).

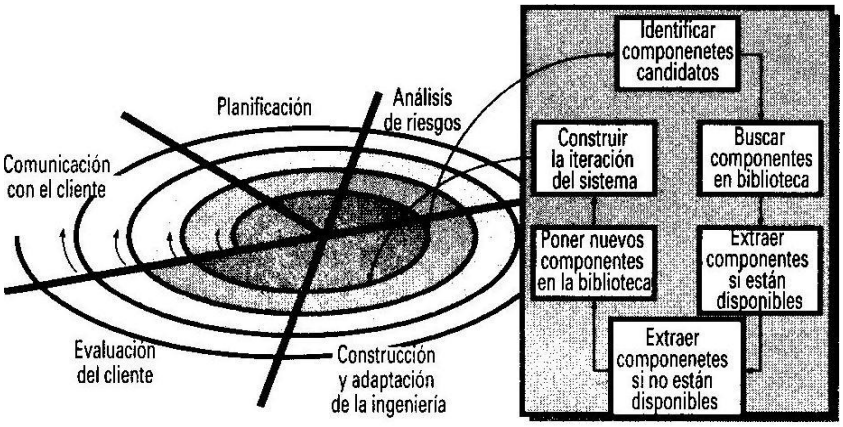


Figura 7. Arquitectura Basada en Componentes.

El modelo de desarrollo basado en componentes incorpora muchas de las características del modelo en espiral. Es evolutivo por naturaleza y exige un enfoque iterativo para la creación del software. Sin embargo, el modelo de desarrollo basado en componentes configura aplicaciones desde componentes preparados de software (llamados «clases»). El modelo de desarrollo basado en componentes conduce a la reutilización del software, y la reutilización proporciona beneficios a los ingenieros de software. Según estudios de reutilización, *QSM*[17] Associates, Inc. Informa que el ensamblaje de componentes lleva a una reducción del 70 por 100 de tiempo de ciclo de desarrollo, un 84 por 100 del coste del proyecto y un índice de productividad del 26.2, comparado con la norma de industria del 16.9. Aunque estos resultados están en función de la robustez de la biblioteca de componentes, no hay duda de que el ensamblaje de componentes proporciona ventajas significativas para los ingenieros de software (Guio, 2016).

Se elige esta arquitectura debido a que proporciona escalabilidad al sistema, la localización de errores y mejora considerablemente el soporte del mismo. Las interacciones entre estos componentes ocurren por invocación de métodos. Ofrece mayor flexibilidad ya que se pueden añadir nuevos componentes para dotar la aplicación de nuevas funcionalidades. Esta arquitectura nos lleva a alcanzar un mayor nivel de reutilización de software, permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

[17] **QSM** o Gestión de Software Cuantitativo es una empresa dedicada al campo de la estimación de software desde 1978.

2.6.2 Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos, y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí (Larman, 2004).

Patrones de Asignación de Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2004).

Experto: este patrón define como asignar de forma adecuada las responsabilidades en un modelo de clases. Este indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer en la clase que conoce toda la información necesaria para crearlo. Entre sus beneficios se encuentra que permite obtener un diseño con mayor cohesión y conservar el encapsulamiento (Larman, 2004). En el marco de la presente investigación siguiendo el patrón Experto en Información se le asignaron responsabilidades determinadas solamente a las clases que cuentan con la información necesaria para dar cumplimiento a las mismas. Esto se evidencia en la Tabla 5 . De esta forma se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener.

Tabla 5. Relación de asignación de responsabilidades

Clase objeto	Responsabilidad
uppload_notice	Conocer si una noticia fue añadida o no a la Base de Datos.
method_fecha	Conocer la fecha del comentario.
get_dataset	Conocer si los ficheros de la base de conocimiento se encuentran en la dirección física establecida.
pre_processing	Conocer si el texto a analizar fue procesado correctamente.
Miner	Conoce todas las características de las noticias y sus comentarios.

Creador: este patrón guía la asignación de responsabilidades de la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que deba conectar con el objeto producido en cualquier evento. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la

clase creadora (Larman, 2004). La clase “Miner” es la encargada de analizar todo el contenido de las noticias y los comentarios, además de realizar la clasificación, para esto crea instancias de las clases “uppload_notice”, “get_dataset” y “pre_processing”, estas realizan el procesamiento necesario para detectar si la noticia fue añadida correctamente, los ficheros de la base de conocimiento se cargaron y si el texto fue procesado correctamente. La Figura 8 muestra el flujo de creación de los objetos.

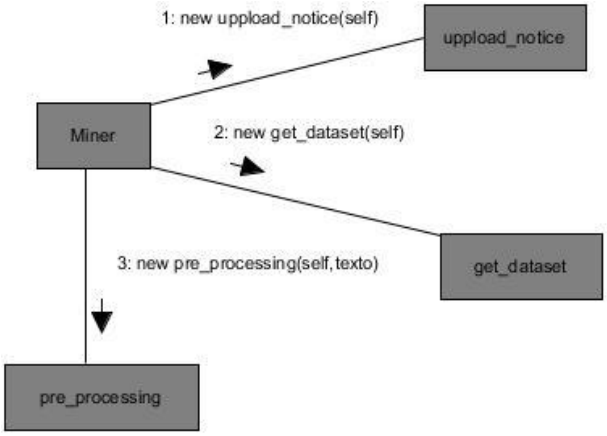


Figura 8. Creación de objetos en la clase Miner

Alta cohesión: el patrón alta cohesión plantea que la información que almacena una clase debe de ser coherente y debe estar en la medida de lo posible relacionada con la clase. En el diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento (clase o subsistema). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Una clase con alta cohesión es relativamente fácil de mantener, entender y reutilizar (Larman, 2004). La utilización de este patrón de diseño se manifiesta en el diseño de la clase “Miner”. Esta clase en orden de cumplir la responsabilidad que le es asignada mediante el patrón Experto, colabora y a su vez delega responsabilidades en las clases “Preprocessing” y “Model”.

Controlador: este patrón es utilizado como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación lo que posibilita aumentar la reutilización de código y a la vez tener un mayor control (Larman, 2004). La utilización de este patrón se evidencia en la clase “ApplicationController”, la misma se

encarga de atender y ofrecer respuesta a cada una de las peticiones realizadas por el usuario mediante la interfaz web.

Patrones GOF[18]

Los patrones GOF que se utilizaron en el desarrollo del Sistema de clasificación de opiniones para medios de prensa digital son:

- **Patrones de Creación:** El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
    1- ***Singleton* (Instancia única):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (Larman, 2004). Este patrón se emplea en la clase "Model" para acceder a todas las funcionalidades de la Base de Datos de manera global. Gracias a este patrón se puede acceder a la clase desde cualquier otro componente del sistema mediante la creación de un objeto de tipo singleton. En la Figura 9 se puede observar cómo se crea el objeto y en la figura 10 como se utiliza el mismo.

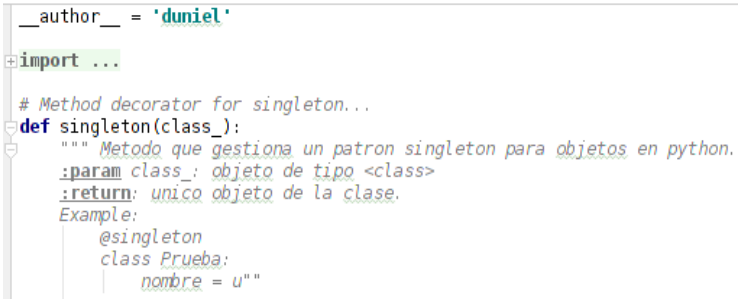


Figura 9. Creación del patrón Singleton

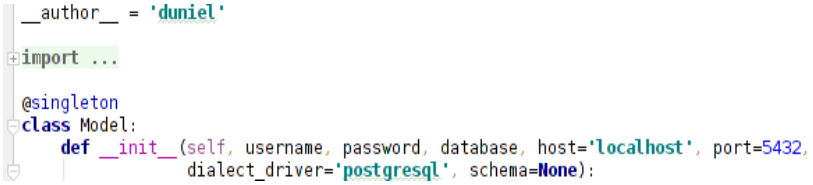


Figura 10. Utilización del patrón Singleton

[18] Gang Of Four. Corresponden a patrones de diseño software que solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.

2.7 Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en el despliegue del sistema y las relaciones entre sus componentes. Muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos (Pressman y Troya, 2010).

Como se puede apreciar en la Figura 11, el nodo “PC Client” representa la computadora utilizada por el usuario desde la cual podrá realizar la clasificación de los comentarios, a través del protocolo HTTP o HTTPS y del puerto 8080, haciendo uso de un navegador web. El nodo “Web Server” se encarga de enviar la solicitud al nodo “Application Server” y este a su vez se encarga de atender y ofrecer respuesta a cada una de las solicitudes del cliente. También se observa un tercer nodo llamado “Databases Server” que es el encargado del tráfico de información entre el sistema y la Base de Datos, la conexión a dicho servidor se realiza mediante los protocolos TCP/IP, por el puerto 5432.

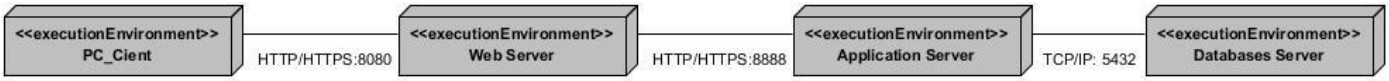


Figura 11. Diagrama de despliegue

A continuación se describen los requisitos que debe cumplir cada nodo:

PC_Client: Este nodo representa la computadora que va a acceder al servicio de clasificación de comentarios. Para consumir dicho servicio debe estar conectada a la red y tener instalado un navegador web ya que mediante este es que puede realizar dicho proceso. Debe tener además el Cntlm configurado con acceso a internet para poder descargar las noticias y sus comentarios.

Web Server: En este nodo se encuentra alojado el servidor Apache 2 que es el encargado de recibir las peticiones del usuario y una vez extraidos los comentarios y la noticia de la red envía los datos hacia el servidor de aplicaciones. En este nodo se debe encontrar instalado las siguientes herramientas y módulos:

- PHP 7.0.18, Apache 2 como servidor web, php7-curl, php7-pgsql para el tratamiento de las consultas sql, php7-dev, php7-cli y php7-common.

***Application Server*:** En este nodo se van a encontrar alojado el servidor de aplicaciones Tornado que será el encargado de procesar todas las consultas enviadas por el servidor web y dar una respuesta al usuario. Deben estar instalados en este las siguientes herramientas y módulos:

- Python 3, python3-pgsql para manejar las consultas sql desde python, python3-tornado como servidor de aplicaciones, python3-nltk y python3-sklearn para el procesamiento de lenguaje

natural, python3-psycopg2, python3-sqlalchemy, python3-setuptools, python3-requests, python3-magic, python3-numpy, python3-scipy, poppler-utils y python 2.7.

Databases Server: Este nodo es el encargado de almacenar y dar al usuario toda la información que se procesa y debe tener instalado el SGBD PostgreSQL 9.4. Garantiza dar una respuesta a cada una de las peticiones que realiza el nodo donde se encuentran alojados los servidores para posteriormente darselas a los usuarios.

2.8 Diagrama de clases del diseño

Un diagrama de clases en UML es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos. El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal. Con el fin de facilitar la comprensión del diagrama, se pueden incluir paquetes como elementos del mismo, donde cada uno de ellos agrupa un conjunto de clases (Ibiblio, 2014).

Para tener una mejor organización de las clases que conforman el Sistema de clasificación de opiniones para medios de prensa, se decidió diseñar el siguiente diagrama de paquetes (ver figura 12) el cual está compuesto por dos sub-paquetes, el primero es *Web_Server*, donde en él se encuentran las clases que permiten el manejo y control del Servidor de la aplicación (ver figura 13), y el segundo es nombrado algoritmo de clasificación que sería el corazón del sistema, es aquí donde se encuentran las clases que rigen el proceso de clasificación de los comentarios en los medios de prensa (ver figura 14).

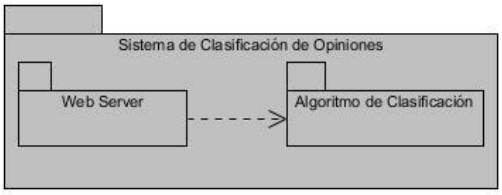


Figura 12. Diagrama de Clases

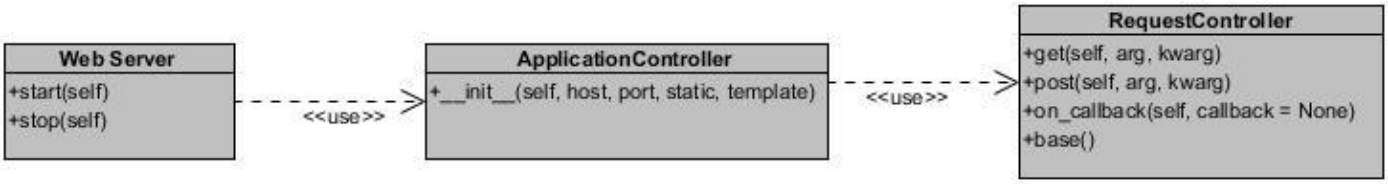


Figura 13. Diagrama de clases perteneciente al sub-paquete Web_Server

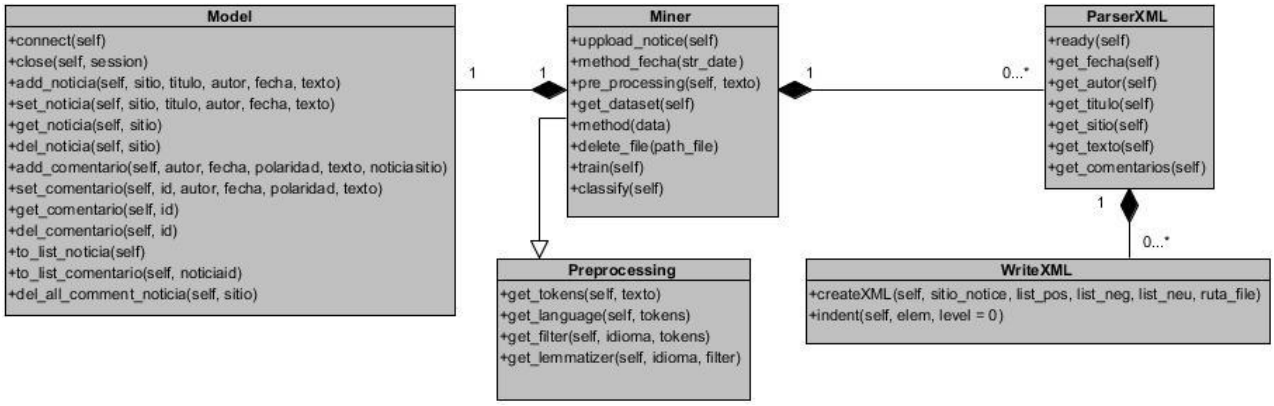


Figura 14.Diagrama de clases perteneciente al sub-paquete Algoritmo de Clasificación

2.9 Conclusiones parciales

- Con el desarrollo de este capítulo se logró especificar las características fundamentales de la propuesta a travéz del modelado de software.
- Se definieron las historias de usuarios que caracterizan la solución, además se especificaron los requisitos funcionales y no funcionales, los cuales proponen las condiciones que se deben tener en cuenta para desarrollar el sistema.
- Se realizó un análisis de los patrones de diseño que fueron utilizados en la aplicación y se pudo obtener una visión para definir la arquitectura, además se confeccionaron otros artefactos por ejemplo, los diagramas de clase del diseño web.

Capítulo 3: Implementación y pruebas del sistema de clasificación de opiniones para medios de prensa digitales

Luego de realizar el diseño del sistema propuesto y de definir su arquitectura, se procede a implementar un software que cumpla con los requisitos que se necesitan. El presente capítulo está dedicado a detallar las dos iteraciones llevadas a cabo durante la etapa de construcción de la aplicación, mostrando los principales artefactos de la implementación. Se presenta la definición de los estándares de codificación que debe tener en cuenta el desarrollador durante la etapa de implementación del software, se muestra una descripción de las pruebas empleadas en la validación de la herramienta desarrollada, presentándose a su vez los resultados obtenidos durante dicho proceso.

3.1 Diagrama de componentes

Los Diagramas de Componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes, por lo cual cada diagrama describe un apartado del sistema (Alvarado y Rodríguez, 2012).

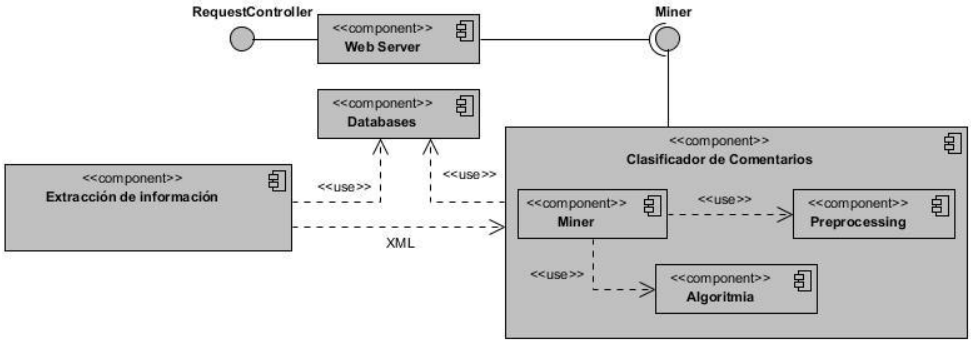


Figura 15. Diagrama de Componentes

A continuación se describen los componentes presentes en el diagrama:

- ***Web_Servers:*** Este componente es el encargado de controlar el funcionamiento general de la aplicación mediante métodos de entrada/salida de información.

- ***Clasificador de comentarios:*** Este componente incluirá en si otros componentes que son los encargados del procesamiento de los comentarios. En este se lleva a cabo la clasificación de las opiniones y se obtiene una lista de los resultados del procesamiento.
- ***Preprocessing:*** Este componente es el encargado de analizar toda la información proveniente del archivo XML o de la base de datos y de construir el vector con esa información.
- ***Miner:*** Es el componente encargado del entrenamiento del modelo y de comenzar y detener el servicio de minería de datos.
- ***Algoritmia:*** Este componente se encarga del análisis del vector y de devolver una respuesta al componente *Miner*.
- ***Extracción de información:*** Es en encargado de extraer toda la información proveniente de las noticias, como son los comentarios, texto de la noticia, url, entre otros.
- ***Databases:*** Es el componente encargado de almacenar toda la información obtenida en la fase de extracción y de clasificación.

3.2 Estándares de codificación

Un estándar de codificación completo, comprende todos los aspectos de la generación del código. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto, trabajen de forma coordinada. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad, es establecer un estándar de codificación, sobre el que se efectuarán luego, revisiones del código (Microsoft, 2014).

Tabla 6. Descripción de los estándares de codificación.

Estilos	Descripción
Pascal	La primera letra en el identificador y la primera letra de cada subsiguiente palabra conectada se capitalizan. Se utiliza en caso de identificadores con tres o más caracteres. Por ejemplo: *RequestController*.
Camello	La primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra conectada en mayúscula. Por ejemplo: *startService*.
Mayúscula	Todas las letras en el identificador se capitalizan. Esta convención se utilizara solo para identificadores que constan de dos o menos letras. Por ejemplo: *System.Text.*

3.2.1 Reglas de codificación

- Se debe evitar las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas herramientas.
- Cada línea debe contener cuando más una sentencia.
- Cada funcionalidad debe tener comentario de su funcionamiento.
- Se debe seguir las convenciones de nombre mostradas en la tabla 7.

Tabla 7. Convenciones de nombres.

Tipos de Identificadores	Reglas de nombre	Ejemplo
Clases o Interfaces.	Los nombres de las clases o interfaces debe tener la primera letra de cada palabra en mayúscula.	Opinion, SistemaClasificación
Métodos.	Los nombres de los métodos deben reflejar la acción a realizar y siempre comenzando con letra mayúscula, en caso de ser compuesto ambas palabras en mayúscula.	ClassifierOpinion, GetLemmatizer.
Variables.	Todas las variables empezarán con minúscula y la primera letra de las siguientes palabras en minúscula.	Identificador, entrenarred.
Constantes.	Cada carácter que pertenezca al nombre de la constante se escribirá en mayúscula y en caso de ser un nombre compuesto, cada palabra se separará por un guión bajo "_".	PI, CARÁCTER_VOID

3.3 Validación de la propuesta de solución

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo (García, 2014). Con el fin de validar el correcto funcionamiento del Sistema de clasificación de opiniones para medios de prensa digitales se realizan pruebas de funcionalidad, integración y seguridad. A continuación se describe el proceso necesario para llevar a cabo dichas pruebas.

3.3.1 Pruebas funcionales

Las pruebas de funcionalidad se enfocan en las acciones por parte del usuario y las respuestas por parte del sistema, se llevan a cabo para comprobar los requerimientos y se considera que una funcionalidad

tiene éxito cuando se comporta de la manera esperada por el cliente (García, 2014). Para comprobar el cumplimiento de las funcionalidades del sistema se diseñaron casos de prueba para cada HU. A continuación se relacionan las más importantes.

Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Caso de Prueba 1: SC RF1_Cargar noticias.

Descripción general: El sistema permite cargar las noticias desde los medios de prensa.

Condiciones de ejecución: El sistema debe estar conectado a la red.

Tabla 8. Caso de prueba 1: Cargar noticias.

Escenario	Descripción	Noticias	Respuesta del sistema	Flujo central
EC 1.1 Cargar la noticia con sus comentarios desde una URL válida.	El sistema carga las noticias desde los medios de prensa de forma correcta.	V Noticia Existente	El sistema carga las noticias y las almacena en la BD para ser analizadas.	El usuario introduce la URL de la noticia a analizar y oprime el botón "Cargar".
EC 1.2 Cargar la noticia con sus comentarios desde una URL inválida.	El sistema no carga las noticias desde los medios de prensa de forma correcta.	I Noticia no existente	El sistema no carga las noticias, no almacena ningún valor en la BD.	El usuario introduce la URL de la noticia a analizar y oprime el botón "Cargar".
EC 1.3 Cargar la noticia con sus comentarios desde una URL dejando el campo vacío.	El sistema no realiza ninguna acción debido a que el campo de entrada (URL) tiene que ser obligatorio.	NA	El sistema no realiza ninguna acción ya que debe contener una URL para ejecutarse.	El usuario no introduce la URL a analizar y oprime el botón "Cargar".

Descripción de las variables:

Tabla 9. Descripción de las variables del caso de prueba 1.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	URL	campo de texto	No	Caracteres alfanuméricos.

Caso de Prueba 2: SC RF2_Clasificar comentarios.

Descripción general: El sistema permite clasificar los comentarios en positivos o negativos.

Condiciones de ejecución: El sistema debe haber realizado el pre-procesamiento de los datos y haber guardado en la BD el resultado de la clasificación.

Tabla 10. Caso de prueba 2: Clasificar comentarios.

Escenario	Descripción	Elementos	Respuesta del sistema	Flujo central
EC 2.1 Clasificar el comentario de forma positiva.	El sistema clasifica los comentarios de forma positiva.	V Comentarios	El sistema clasifica los comentarios de manera positiva y lo añade en una lista.	El usuario selecciona la opción clasificar comentarios y oprime el botón "Clasificar".
EC 2.2 Clasificar comentarios de forma positiva.	El sistema clasifica los comentarios de forma positiva mediante los falsos positivos	I Comentarios	El sistema clasifica los comentarios de manera positiva dando resultados no esperados.	
EC 2.3 Clasificar comentarios de forma positiva.	El sistema clasifica los comentarios de forma positiva mediante los falsos negativos.	I Comentarios	El sistema clasifica los comentarios de manera positiva cuando deberían ser negativos.	
EC 2.4 Clasificar comentarios de forma negativa.	El sistema clasifica los comentarios de forma negativa.	V Comentarios	El sistema clasifica los comentarios de manera negativa y los añade a una lista.	

Descripción de las variables.

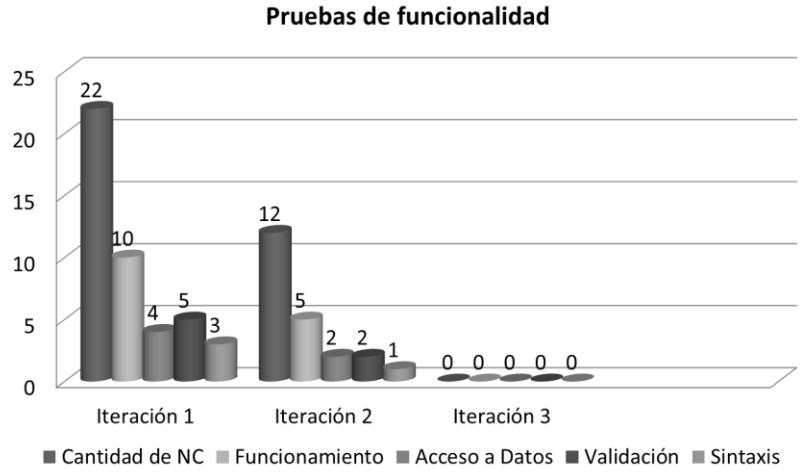
Tabla 11. Descripción de las variables del caso de prueba 2.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Elementos	Campo de acción (Botón)	no	Realiza la clasificación de los comentarios en positivos o negativos.

Resultados de las pruebas de funcionalidad

En la primera iteración se detectan 22 no conformidades, de ellas 6 de validación, 8 son de error de concordancia, 5 de ortografía y 3 de recomendación. En la segunda iteración se encuentran un total de 12 no conformidades 5 de validación, 4 de error de concordancia, 2 de ortografía y 1 de recomendación. Se realizó una tercera iteración en la que no se encontraron no conformidades. Como resultado final todas las No conformidades encontradas fueron resueltas, lo que demuestra que el software funciona correctamente.

Figura 16. Resultados de las pruebas de Funcionalidad.



3.3.2 Pruebas de seguridad

La seguridad informática comprende la puesta en práctica de un conjunto de medidas preventivas y reactivas en los sistemas informáticos y tecnológicos, que posibilitan la protección de la información, persiguiendo como objetivo principal la integridad, confidencialidad y disponibilidad de la misma (Quality, 2014).

La realización de pruebas de seguridad contribuye a la detección temprana de vulnerabilidades y la toma de medidas para la disminución de amenazas de ataque, y con ello proveer sistemas de cómputo más seguros y confiables. A la herramienta desarrollada se le realizaron una serie de pruebas de seguridad mediante el software Acunetix, las cuales se presentan a continuación:

- Ataques de inyección, *Cross-Site Scripting* (XSS), Falsificación de petición (CSRF), Detección de ficheros y directorios.

Se encontraron 16 alertas en total, de ellas ninguna de clasificación alta, 3 de clasificación media, 4 de clasificación baja y 9 informacionales. De manera general los resultados obtenidos se presentan en la Tabla 12.

Tabla 12: Vulnerabilidades del sistema.

Tipo	Cantidad	Descripción	Recomendaciones
Software	1	La versión de Python instalada contiene vulnerabilidades que han sido solucionadas en	Actualizar a la versión 2016.3 o superior.

		versiones más recientes.	
Configuración	13	La configuración del servidor permite la captación de información sensible por el atacante.	Esta vulnerabilidad aunque no permite al atacante tomar el control acerca de la información a obtener puede proporcionar al atacante información sensible que puede utilizar para realizar ataques más específicos, pueden solucionarse limitando el acceso a los archivos del servidor y los elementos de la configuración ajustando el "cinfig_init" en “off” en el archivo configuracion.ini.
Falso positivo	1	El sistema permite la inyección de código.	Esta vulnerabilidad no puede ser utilizada por el atacante para alterar el sistema ni obtener información por lo que se denomina falso positivo.

Observaciones: Todas las vulnerabilidades fueron resueltas en el servidor y se recomienda que se tenga en cuenta la nueva configuración en el servidor para un futuro despliegue del sistema.

Los resultados obtenidos en las pruebas de seguridad, avalados por el centro de calidad UCI, fueron satisfactorios, llegando a la conclusión que la herramienta desarrollada es segura y está en condiciones de ser usada por el cliente con el propósito de satisfacer las necesidades de clasificación de comentarios en los medios de prensa digitales.

3.3.3 Pruebas de carga y estrés

Las pruebas de carga consisten en probar el funcionamiento del software bajo condiciones extremas. Estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas, analizando los valores límites (García, 2014). Las pruebas de estrés están diseñadas para enfrentar al programa a condiciones anormales. Las pruebas ejecutan un sistema, de manera que demande recursos en cantidad, frecuencia o volúmenes extremos. Para la herramienta de clasificación de opiniones en medios de prensa digitales es preciso realizar dichas pruebas pues resulta necesario comprobar el rendimiento del sistema soportando una cantidad máxima de usuarios que soliciten este recurso en la Web. Para la realización de las mismas se utiliza la herramienta JMeter en su versión 3.1.

La prueba realizada consistió en definir 2 pruebas de 200 y 300 hilos de concurrencia cada una, las cuales simulan 200 y 300 accesos de usuarios respectivamente. La primera prueba fue realizada en un servidor Pentium(R) Dual-Core a 2.20Ghz de velocidad de procesador, una memoria RAM de 4.00Gb y a las 21:07

hs para 200 hilos, cada “1 seg”. La segunda prueba fue realizada en una PC Core-i3 a 2.4Ghz de velocidad de procesador, una memoria RAM de 6.00Gb, a las 12:30hs configurando 300 hilos, cada “1 seg”. Para un mejor entendimiento de las componentes “Reporte resumen” que se verán a continuación, se explica cada parámetro que la compone.

#Muestras: cantidad de hilos utilizados para la URL.

Media: tiempo promedio en milisegundos para un conjunto de resultados.

Min: tiempo mínimo que demora un hilo en acceder a una página.

Max: tiempo máximo que demora un hilo en acceder a una página.

Rendimiento: rendimiento medido en los requerimientos por segundo/minuto/hora.

Kb/sec: rendimiento medido en Kbytes por segundo.

Los valores totales obtenidos por la componente “Reporte resumen” para 200 hilos se muestran en la Tabla 13.

Tabla 13: Resultados de las prueba de carga y estrés con el acceso de 200 usuarios.

Total	#Muestras	Media	Min	Max	%Error	Rendimiento	Kb/sec
	5425	2792	1	65170	0	31.683/segundos	2602.274

Como se muestra en la Tabla 13, la herramienta desarrollada, para 200 usuarios conectados de forma concurrente respondió 5 425 peticiones al servidor en un promedio de 2.792 segundos, lo que equivale a 31.683 peticiones por segundo.Los valores totales obtenidos por la componente “Reporte resumen” para 300 hilos se muestran en la Tabla 14.

Tabla 14 Resultados de las prueba de carga y estrés con el acceso de 300 usuarios.

Total	#Muestras	Media	Min	Max	%Error	Rendimiento	Kb/sec
	10200	4605	2	132957	0.029	47.2/segundos	6659.84

Como se muestra en la Tabla 14, la herramienta desarrollada, para un total de 300 usuarios conectados de forma concurrente respondió 10 200 peticiones al servidor en un promedio de 4.605 segundos, lo que equivale a 47.2 peticiones por segundo. De manera general estos resultados son favorables, siendo este satisfactorio para el autor de la presente investigación.

3.3.4 Validación de la hipótesis de la investigación

Con el propósito de evaluar los indicadores referentes al comportamiento de las variables dependientes: “tiempo que se demora en conocer el estado de opinión de los lectores”, “alta precisión”, “escalabilidad del

sistema” y de determinar si la hipótesis de la presente investigación es apoyada o refutada de acuerdo a los resultados obtenidos con el desarrollo del subsistema propuesto, se realizó un experimento donde se compararon los resultados obtenidos por el proceso de clasificación de los comentarios de forma manual y en la herramienta implementada.

Análisis de la variable dependiente: “tiempo que se demora en conocer el estado de opinión de los lectores”

Para el experimento se crearon tres archivos con 100 comentarios cada uno, donde de los 100, 50 fueron clasificadas como positivas, 30 como negativas y 20 como neutras. Para evaluar el indicador **tiempo de demora** se realizaron 3 iteraciones de pruebas con diferentes criterios de información, entre ellos el tiempo que demora clasificar 100, 200 y 300 comentarios respectivamente. En un primer momento, se realizó la clasificación de los comentarios de forma manual sin asignarle el sistema de clasificación de opiniones desarrollado y luego se realizó nuevamente la clasificación pero con dicho sistema integrado.
En la siguiente tabla se muestran los resultados del experimento realizado.

Tabla 15. Resultados de la medición de la variable " tiempo que se demora en conocer el estado de opinión de los lectores "(en segundos).

Cantidad de comentarios		**100**	**200**	**300**
Modo de clasificación	Sin el sistema	7 235.23	16 250.50	21 620.10
	Con el sistema	12.238	14.457	16.970

A partir de la comparación de los tiempos medios obtenidos en el experimento realizado (Ver tabla 15), se evidencia una reducción significativa del tiempo empleado por un usuario al conocer el estado de opinión de los lectores con el uso del sistema desarrollado. Por tal motivo, la hipótesis de investigación anteriormente planteada es apoyada para la variable “tiempo que se demora en conocer el estado de opinión de los lectores”.

Análisis de la variable dependiente: “alta precisión”

Para evaluar los indicadores referentes a la precisión general del sistema se analizaron tres conjuntos de datos con diferentes características. El primero hace alusión a una noticia escrita por nuestro invencible comandante en jefe Fidel Castro Rúz sobre su cumpleaños 90, donde narra los sucesos de cuando era niño y sus primeras actividades revolucionarias en Cuba. Luego de analizar todos los comentarios referentes al tema y clasificar manualmente cada uno de ellos se obtuvieron un total de 335 comentarios

positivos, 11 negativos y 14 neutrales. A continuación se muestra el resultado (en porciento) de la precisión del sistema para este conjunto de datos (ver Tabla 16).

Tabla 16. Precisión del sistema para el primer conjunto de datos.

Clasificación	Cantidad de comentarios	Comentarios bien clasificados	Precisión %
positivos	335	314	**89.16**
negativos	11	3	
neutral	14	4	

El segundo conjunto de datos es referente al deporte en Cuba y trata de la victoria del equipo de Granma en el basebool, además abordan el tema de Victor Mesa como manager de equipo de Industriales para la próxima selección nacional. En la noticia se ve presente un alto nivel de expresión negativa por parte de los usuarios ya que la mayoría de ellos no desean que Victor Mesa esté dirigiendo este equipo. Después de clasificar cada comentario de forma manual se identificaron un total de 30, 260 y 18 comentarios positivos, negativos y neutrales respectivamente. A continuación se muestra el resultado (en porciento) de la precisión del sistema para el segundo conjunto de datos (ver Tabla 17.)

Tabla 17. Precisión del sistema para el segundo conjunto de datos.

Clasificación	Cantidad de comentarios	Comentarios bien clasificados	Precisión %
positivos	30	20	**87.66**
negativos	260	232	
neutral	18	18	

El tercer conjunto de datos analizados trata de una entrevista realizada por Amaury Pérez Vidal al trovador Silvio Rodríguez como clausura de una de las temporadas del programa televisivo “Con 2 que se quieran”. Una vez realizada la clasificación manual de los comentarios referentes a la noticia se obtuvieron un total de 536 comentarios positivos, 470 negativos y 38 neutrales. A continuación se muestra el resultado (en porciento) de la precisión del sistema para el tercer conjunto de datos (ver Tabla 18.)

Tabla 18. Precisión del sistema para el segundo conjunto de datos.

Clasificación	Cantidad de comentarios	Comentarios bien clasificados	Precisión %
positivos	536	490	**85.82**
negativos	470	381	
neutral	38	25	

Teniendo en cuenta la variable dependiente analizada y el porciento de precisión del sistema que se encuentra entre 85.82 y 89.16 se puede decir que la hipótesis anteriormente planteada es apoyada en cuanto a la precisión.

Análisis de la variable dependiente: "escalabilidad del sistema"

Para analizar la escalabilidad del sistema se tuvo en cuenta las actualizaciones y cambios que se le realizaron a cada componente que lo necesitara y en dependencia de la modificación ver si este funcionaba correctamente. En la Tabla 19 se muestran dichos cambios y dice si el componente se logra incorporar de manera correcta.

Tabla 19. Medición de la variable dependiente "escalabilidad del sistema".

Componente a modificar	Descripción del cambio	Incorporación
Extractor de información	Se le añade la funcionalidad de extraer de un texto determinado el autor, el texto y la fecha de un comentario.	Correctamente
Databases	Se modifica el acceso a la base de datos y solamente guarda de las noticias la dirección url, el texto y el tema.	Correctamente
Miner	Se modifica el archivo *dataset* que es el contenedor de la base de conocimientos y se añade a esta comentarios en vez de palabras.	Correctamente
Analitic	Se elimina la funcionalidad de iniciar el servidor automáticamente.	Correctamente

La incorporación exitosa, modificación y actualización de los diversos componentes que integran el sistema permite que este sea escalable, dándole la posibilidad que puedan surgir nuevas versiones de los componentes e incorporarse sin ninguna deficiencia. Lo anteriormente planteado apoya la hipótesis en cuanto a escalabilidad.

Conclusiones de la validación de la hipótesis

Las pruebas realizadas anteriormente apoyan la hipótesis de la investigación dando como resultados que el sistema desarrollado presenta un buen tiempo para conocer el estado de opinión de los lectores con una alta precisión y logrando que este sea escalable, lo que cumple con el objetivo de esta investigación.

3.4 Conclusiones del capítulo

- En este capítulo, se generaron algunos de los principales artefactos de la metodología AUP-UCI.
- Se modeló el diagrama de componente y despliegue del sistema, con el objetivo de lograr una mayor comprensión y organización de la misma.
- Se valida el sistema desarrollado, realizando casos de prueba a todas las funcionalidades del mismo en tres iteraciones. Este proceso permite detectar la mayor cantidad de no conformidades presentes en el software, para así darle solución a los problemas detectados. De esta manera, es posible obtener finalmente un sistema que satisface los requisitos definidos y que cuenta con la calidad requerida.

Conclusiones

Luego de la realización de la investigación pertinente y el desarrollo del Sistema de clasificación de opiniones para medios de prensa digitales se llega a las siguientes conclusiones:

- ✓ A partir del estudio realizado de los fundamentos teóricos relacionados con la clasificación de opiniones en el procesamiento de lenguaje natural se determinó que existen una serie de Sistemas de Clasificación de Textos los cuales brindan funcionalidades que implican un procesamiento previo de los textos, no encontrándose ninguna de código abierto para su uso, definiéndose una propuesta de solución de acuerdo a las necesidades existentes.
- ✓ Una vez estudiados los elementos que intervienen en el proceso de clasificación de texto, fue posible la modelación de los artefactos que contribuyeron al diseño de la propuesta de solución posibilitando un mayor soporte a la implementación de los requisitos previamente expresados por el cliente; garantizando la estructura base para la organización lógica del código fuente y la disminución del impacto ante futuras modificaciones en la aplicación.
- ✓ La implementación de la herramienta informática para la clasificación de texto utilizando las herramientas y tecnologías estudiadas y orientada por las fases de la metodología AUP-UCI permitió solucionar los problemas existentes planteados en la problemática de la presente investigación.
- ✓ Las pruebas de software realizadas al sistema permitieron mejorar la calidad de la aplicación.

Recomendaciones:

- Implementar un módulo que sea capaz de clasificar los comentarios de manera temática y adicionárselo al sistema implementado.

Bibliografía

ACUNA, E. *Clasificadores Naive Bayes* [en línea]. 2015. S.l.: s.n. Disponible en: http://academic.uprm.edu/eacuna/naivebayesacu.pdf.

ADSENSE Filtrado de contenido. *Adsense* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: https://support.google.com/adsense/answer/3011871?hl=es.

ALVARADO TRUJILLO, F. y RODRÍGUEZ MORERA, M. Diagramas de Componentes. *prezi.com* [en línea]. 2012 [Consulta: 25 junio 2017]. Disponible en: https://prezi.com/yzvn1klhsd4x/diagramas-de-componentes/.

ÁLVAREZ, M.Á. Qué es Python. *DesarrolloWeb.com* [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: http://www.desarrolloweb.com/articulos/1325.php.

AMAYA, V. ¿Qué hacer para dominar el arte del levantamiento de requerimientos? [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/RevistaSG/webinar-levantamiento-reqsv1.

AMORES, M., ARCO, L. y ARTILES, M. PosNeg opinion: Una herramienta para gestionar comentarios de la Web. *Revista Cubana de Ciencias Informáticas*, 2015, vol. 9, no. 1, pp. 20-31. ISSN 2227-1899.

ANDREARRS. PyCharm el IDE de Python. *Hipertextual* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: https://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

ANTIGUA, J. Perspective, la herramienta inteligente de Google. *Gikplus* [en línea]. 2017, [Consulta: 25 junio 2017]. Disponible en: http://gikplus.com/2017/02/23/perspective-la-herramienta-inteligente-google/.

ASTUDILLO, C., SQUADRITO, K., VARAS, G., GONZÁLEZ, C. y SABAJ, O. Polaridad de los comentarios y consistencia interna en los informes de arbitraje de artículos de investigación. *Acta bioethica*, 2016, vol. 22, no. 1, pp. 1-10.

BERNARDO QUINTERO, J. Arquitectura de Software. Definiciones y Contexto. [en línea]. Ingeniería de Software. Colombia. 2010, [Consulta: 25 junio 2017]. Disponible en: http://aprendeenlinea.udea.edu.co/lms/moodle/pluginfile.php/109854/mod_resource/content/0/Presentaciones/1-Arquitectura_de_Software.pdf.

BRAVO CASTRO, R.P. y RUILOVA ROJAS, M.E. Árboles de clasificación (Inteligencia Artificial Avanzada). [en línea]. Tecnología. S.l.: Universidad Técnica Particular de Loja. 2015, [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/techi322/algoritmos-de-clasificacin.

CAMPOS CHIU, C. *Las pruebas en el desarrollo de software.* [en línea]. 2015, S.l.: s.n. Disponible en: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/7627/Las%20pruebas%20en%20el%20desarrollo%20de%20software.pdf?sequence=1.

CÁRDENAS, J., OLIVARES, G. y ALFARO, R. Clasificación automática de textos usando redes de palabras. *Revista signos*, 2014, vol. 47, no. 86, pp. 346-364. ISSN 0718-0934. DOI 10.4067/S0718-09342014000300001.

CARMONA SUÁREZ, E.J. Tutorial sobre máquinas de vectores soporte (sVM). *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, 2014, vol. 6, no. 1, pp. 25.

CASTILLO, J., CARDENAS, M. y CURTI, A. Creación de corpus para aplicaciones de análisis de texto no estructurado. En: Facultad Regional Córdova, *corpus*, 2017, vol. 9, pp. 2-9.

CASTRO GALLARDO, J. *Un nuevo modelo ponderado para Sistemas de Recomendación Basados en Contenido con medidas de contingencia y entropıa* [en línea]. Trabajo Tutelado de Iniciación a la Investigación. Jaén: Universidad de Jaén. 2012, [Consulta: 25 junio 2017]. Disponible en: http://sinbad2.ujaen.es/sites/default/files/publications/TTII_JorgeCastro.pdf.

DATACENTER, S. ¿Qué es Aprendizaje automático (machine learning)? - Definición en WhatIs.com. [en línea]. 2017, [Consulta: 24 junio 2017]. Disponible en: http://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-automatico-machine-learning.

DIEDERICH, J., KINDERMANN, J., LEOPOLD, E. y PAASS, G. Authorship Attribution with Support Vector Machines. *Applied Intelligence*, 2012, vol. 19, no. 1-2, pp. 109–123. ISSN 0924-669X. DOI 10.1023/A:1023824908771.

DUDA, R. o., HART, P.E. y STORK, D.G. *Pattern Classification* [en línea]. Second Edition. New York: Wiley Authenticity Guarantee. 2001, [Consulta: 25 junio 2017]. ISBN 978-0-471-05669-0. Disponible en: http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471056693.html.

DUFORT, G., KREMER, Á.F. y MORDECKI, G. *Determinación de la orientación semántica de las opiniones transmitidas en textos de prensa* [en línea]. Proyecto de grado Ingeniería en Computación. Montevideo, Uruguay: Universidad de la República. 2016, [Consulta: 25 junio 2017]. Disponible en: https://www.fing.edu.uy/inco/grupos/pln/prygrado/informe_analisis_sentimientos_2015.pdf.

EIKVIL, L. *Information Extraction from World Wide Web.* [en línea]. 2016. S.l.: s.n. [Consulta: 25 junio 2017]. Disponible en: http://wwwusers.di.uniroma1.it/~estrinfo/IEsurvey.pdf.

ESULI, A. y SEBASTIANI, F. Determining Term Subjectivity and Term Orientation for Opinion Mining. *EACL* [en línea]. S.l.: s.n., pp. 2006. [Consulta: 25 junio 2017]. Disponible en: http://www.esuli.it/publications/EACL2006.pdf.

FABIAN NAZAR, L.A. Uml tutorial-visual-paradigm. [en línea]. Software. 2014, S.l. [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/fabiannazar1/uml-tutorialvisualparadigm.

FINN, A., KUSHMERICK, N. y SMYTH, B. Genre classification and domain transfer for information filtering. *ECIR*. 2002, S.l.: Springer, pp. 353-362.

GALARZA, C. INVESTIGACION DE CIENCIA Y TECNOLOGIA: LAS BASES DEL CONOCIMIENTO - ACIERTOS, ERRORES Y FALACIAS. *INVESTIGACION DE CIENCIA Y TECNOLOGIA* [en línea]. 2015, [Consulta: 25 junio 2017]. Disponible en: http://cienciaytecnologiapanchos.blogspot.com/2014/10/las-bases-del-conocimiento-aciertos.html.

GARCÍA, F. Entorno de Desarrollo Integrado (IDE). *fergarciac* [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

GARCÍA OTERINO, A.M. del C. ¿Pruebas de integración, funcionales, de carga...? ¡Qué jaleo! ¿Qué diferencias hay? *Javier Garzás* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.

GARCÍA, V. El Lenguaje C#. *Scribd* [en línea]. 2016, [Consulta: 25 junio 2017]. Disponible en: https://es.scribd.com/document/2984731/El-Lenguaje-C.

GIMENO, L. Google desarrolla “Perspective”: una herramienta para la lucha contra los “trolls”. *RCI | Español* [en línea]. 2017, [Consulta: 25 junio 2017]. Disponible en: http://www.rcinet.ca/es/2017/02/23/google-desarrolla-perspective-una-herramienta-para-la-lucha-contra-los-trolls/.

GÓMEZ DÍAZ, R. *La lematización en espanõl: una aplicación para la recuperación de información*. Gijón: Ed. Trea. Biblioteconomía y administración cultural, 2005, 125. ISBN 978-84-9704-186-7.

GÓMEZ QUESADA, F.J., FERNÁNDEZ GRACIANI, M.Á., LÓPEZ BONAL, M.T. y DÍAZ MARTA, M.A. Aprendizaje con redes neuronales artificiales. [en línea]. Científico. S.l.: 2014, [Consulta: 25 junio 2017]. 5. Disponible en: https://previa.uclm.es/ab/educacion/ensayos/pdf/revista9/9_19.pdf.

GONZÁLEZ, A. Conceptos básicos de Machine Learning. [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://cleverdata.io/conceptos-basicos-machine-learning/.

GUIO, J.D. MODELO DE ENSAMBLAJE DE COMPONENTES. [en línea]. S.l. 2016, [Consulta: 25 junio 2017]. Disponible en: https://prezi.com/gweqw7vwkebw/modelo-de-ensamblaje-de-componentes/.

HACKERUNA. Algunas de las mejores herramientas de hacking de 2016 | hackeruna.com. [en línea]. 2016, [Consulta: 25 junio 2017]. Disponible en: http://hackeruna.com/2016/10/07/algunas-de-las-mejores-herramientas-de-hacking-de-2016/.

HERNÁNDEZ HERNÁNDEZ, J.M. *Análisis automático de textos en español utilizando NLTK* [en línea]. Grado en Ingeniería Informática. S.l.: Universidad de lA lAGUNA. 2016, [Consulta: 25 junio 2017]. Disponible en: https://riull.ull.es/xmlui/bitstream/handle/915/3082/Analisis%20automatico%20de%20textos%20en%20espanol%20utilizando%20NLTK.pdf?sequence=1.

HERNÁNDEZ ORALLO, E. El Lenguaje Unificado de Modelado (UML). [en línea]. Ingeniería de Software. S.l.: 2015, [Consulta: 25 junio 2017]. 5. Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF. 25413

HERRERA, Y. Aprendizaje Supervisado y no Supervisado. *Redes Neuronales* [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: http://redesneuronares.blogspot.com/.

IBIBLIO, Análisis y Diseño con el Diagrama de Clase. *Modelado de Sistemas com UML* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://www.ibiblio.org/pub/linux/docs/LuCaS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

ISOCIAL WEB. MYSQL ¿Qué es y para qué sirve? *Posicionamiento web SEO y diseño desarrollo web en Barcelona* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://www.isocialweb.es/mysql-que-es-y-para-que-sirve/, http://www.isocialweb.es/mysql-que-es-y-para-que-sirve/.

KESSLER, B., NUMBERG, G. y SCHÜTZE, H. Automatic detection of text genre. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics* [en línea]. S.l.: Association for Computational Linguistics, pp. 32–38. 2014, [Consulta: 25 junio 2017]. Disponible en: http://dl.acm.org/citation.cfm?id=979622.

LARMAN, C. *Modelo de Dominio* [en línea]. Segunda Edición. Madrid: Pearson Educación. 2004, [Consulta: 25 junio 2017]. 28042, 24045. ISBN 84-205-3438-2. Disponible en: http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf. M.23.390-20047167

LARMAN, C. *UML y Patrones* [en línea]. Segunda Edición. Madrid: Pearson Educación. 2004, [Consulta: 25 junio 2017]. 28042, 24045. ISBN 84-205-3438-2. Disponible en: http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf. M.23.390-20047167

LOPEZ BRIEGA, R.E. Machine Learning con Python. [en línea]. 2015, [Consulta: 25 junio 2017]. Disponible en: http://relopezbriega.github.io/blog/2015/10/10/machine-learning-con-python/.

MATRIAM. Desarrollo de Software basado en componentes. *Matriarm's Blog* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: https://matriarm.wordpress.com/desarrollo-basado-en-componentes/.

MICROSOFT. Revisiones de código y estándares de codificación. *Microsoft Developer Network* [en línea]. 2014 [Consulta: 25 junio 2017]. Disponible en: https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

MINEWISKAN. Cross-Validation (Analysis Services - Data Mining). [en línea]. 2017, [Consulta: 24 junio 2017]. Disponible en: https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/cross-validation-analysis-services-data-mining.

MORA SOTO, J.A. Herramientas UML… ¿cuál utilizar? *Jose Arturo Mora Soto* [en línea]. 2010, [Consulta: 25 junio 2017]. Disponible en: http://www.jams.name/2010/04/18/herramientas-uml-cual-utilizar/.

NIPAS, G. Sistemas gestores de base de datos. [en línea]. Software. México. 2014, [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/nipas/10-sgbd.

PÉREZ AGUILAR, C. Modelado de Datos. [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: http://ict.udlap.mx/people/carlos/is341/bases02.html.

PÉREZ, I. Lenguajes de programación. *CCM* [en línea]. 2017, [Consulta: 25 junio 2017]. Disponible en: http://es.ccm.net/contents/304-lenguajes-de-programacion.

PLASCENCIA, S. Herramientas para Pruebas de Software. *Prezi* [en línea]. 2015, [Consulta: 25 junio 2017]. Disponible en: https://prezi.com/o92ydwg4al5a/herramientas-para-pruebas-de-software/.

PMOINFORMÁTICA. ¿Qué son las historias de usuario? *PMOInformática.com* [en línea]. 2013, [Consulta: 25 junio 2017]. Disponible en: http://www.pmoinformatica.com/2013/04/que-son-las-historias-de-usuario-7.html.

POSTGRESQL. PostgreSQL: Características, limitaciones y ventajas. *PostgreSQL* [en línea]. 2016, [Consulta: 25 junio 2017]. Disponible en: http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

PRESSMAN, R.S. y TROYA, J.M. *Ingeniería del software. Un enfoque práctico.* [en línea]. 7ma Edición. México: Cámara Nacional de la Industria Editorial Mexicana. 2010, [Consulta: 25 junio 2017]. 109876543210. ISBN 978-607-15-0314-5. Disponible en: http://www.academia.edu/download/45525376/Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF. 1234567890

PRESSMAN, R.S. y TROYA, J.M. *Ingeniería del software. Un enfoque práctico.Modelo de despliegue.* [en línea]. 7ma Edición. México: Cámara Nacional de la Industria Editorial Mexicana. 2010, [Consulta: 25 junio 2017]. 109876543210. ISBN 978-607-15-0314-5. Disponible en: http://www.academia.edu/download/45525376/Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF. 1234567890

PRIETA, M. Análisis y clasificación de textos. [en línea]. Educación. S.l. 2013, [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/martinha_prieta/analisis-y-clasificacion-de-textosselectividad.

QUALITY. Pruebas de Seguridad. *Quality Performance* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://vyvquality.com/pruebas-seguridad/.

RAMÍREZ BENAVIDES, K.D. *Stemming-Lematización* [en línea]. 2014. S.l.: UCR – ECCI. Disponible en: http://www.kramirez.net/wp-content/uploads/2012/02/Stemming.pdf.

RODRÍGUEZ SÁNCHEZ, T. *Metodología de desarrollo para la Actividad productiva de la UCI*. 9 diciembre 2014. S.l.: s.n.

SANG, K. Ventajas y Desventajas de MySQL. *prezi.com* [en línea]. 2012, [Consulta: 25 junio 2017]. Disponible en: https://prezi.com/esjdscte98lr/ventajas-y-desventajas-de-mysql/.

SAORI. Manipulación de la información por los medios de comunicación. [en línea]. Educación. S.l. 2012, [Consulta: 24 junio 2017]. Disponible en: https://es.slideshare.net/Saori06/manipulacin-12821966.

SAULA, A. Java y sus caracteristicas. [en línea]. Tecnologías. S.l. 2014, [Consulta: 25 junio 2017]. Disponible en: https://es.slideshare.net/anasaula9/java-y-sus-caracteristicas.

SERVERTSON, B. Limpieza y preparación de datos para Azure Machine Learning. [en línea]. 2017, [Consulta: 25 junio 2017]. Disponible en: https://docs.microsoft.com/es-es/azure/machine-learning/machine-learning-data-science-prepare-data.

SOMMERVILLE, I. *Ingeniería de Software. Somerville* [en línea]. 9na Edición. México: Pearson Educación. [Consulta: 25 junio 2017]. 2011, 978-607-32, 1031. ISBN 978-607-32-0603-7. Disponible en: ftp://april.frm.utn.edu.ar/METODOLOGIA%20DE%20SISTEMAS%20-%20TSP/LIBROS/Ingenieria%20de%20Software-Somerville.pdf. 53519

SPANISH, woodward. Familia de Palabras en español con ejemplos. [en línea]. 2017, [Consulta: 24 junio 2017]. Disponible en: http://www.spanish.cl/vocabulario/familia-de-palabras.htm.

TORRES LÓPEZ, C. y ARCO GARCÍA, L. Representación textual en espacios vectoriales semánticos. *Revista Cubana de Ciencias Informáticas*, 2016, vol. 10, no. 2, pp. 148–180.

TORRES LÓPEZ, C., ARCO GARCÍA, L. y AMORES FERNÁNDEZ, M.A. *Propuesta de incorporación de técnicas de detección de tópicos a PosNeg Opinion* [en línea]. S.l.: s.n. 2015, [Consulta: 25 junio 2017]. Disponible en: https://www.researchgate.net/publication/283463690_Propuesta_de_incorporacion_de_tecnicas_de_deteccion_de_topicos_a_PosNeg_Opinion.

UREÑA, L.A., GARCÍA, M., GÓMEZ, J.M. y DÍAZ, A. Integrando una base de datos léxica y una colección de entrenamiento para la desambiguación del sentido de las palabras. *Procesamiento del Lenguaje Natural*, 2014, vol. 23, pp. 3-7.

UTFSM. Algoritmos. *Universidad Técnica Federico Santa María* [en línea]. 2015, [Consulta: 25 junio 2017]. Disponible en: https://www.inf.utfsm.cl/~contrera/competencias/documentos/AlgoritmoResumen.pdf.

VALLEZ, M. El Procesamiento del Lenguaje Natural en la Recuperación de Información. *núm. 5*, pp. 25-39. 2007.

VÁZQUEZ GARCIA, M. El futuro de las herramientas de procesamiento del lenguaje. *2014-01*, 2014, vol. 29, pp. 1-25. ISSN 2014-2226.

VILARES CALVO, D., PARDO, A., ÁNGEL, M. y GÓMEZ RODRÍGUEZ, C. Clasificación de polaridad en textos con opiniones en español mediante análisis sintáctico de dependencias. [en línea], 2013, [Consulta: 25 junio 2017]. ISSN 1135-5948. Disponible en: http://rua.ua.es/dspace/handle/10045/27859.

WINDSOR, G. Características de GATE. *General Architecture for text engineering* [en línea]. 2016, [Consulta: 25 junio 2017]. Disponible en: https://gate.ac.uk/.

ZATOR. El lenguaje C++. *C++* [en línea]. 2015, [Consulta: 25 junio 2017]. Disponible en: http://www.zator.com/Cpp/E1_2.htm.

ZEBALLOS, Y. *Identificación automática del asunto de opiniones en texto en idioma español.* [en línea]. Informe de Proyecto de Grado. Montevideo, Uruguay: Universidad de la República. 2013. [Consulta: 25 junio 2017]. Disponible en: https://www.fing.edu.uy/inco/grupos/pln/prygrado/InformeOpiniones.pdf.

ZEOKAT. ¿Qué son las stop words o palabras vacías? *Vozidea.com* [en línea]. 2014, [Consulta: 25 junio 2017]. Disponible en: http://www.vozidea.com/que-son-las-stop-words-o-palabras-vacias.