UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Implementación de algoritmos de minería de textos para el análisis de artículos científicos

TESIS PARA OPTAR POR EL GRADO DE

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

ING. LUIS ENRIQUE ARGOTA VEGA

TUTORA:

DRA. MARÍA DEL PILAR ANGELES

Facultad de Ingeniería, UNAM

Ciudad Universitaria, Cd. de México, agosto 2019

DEDICATORIA

A mi hija que le sirva de ejemplo, sacrificio, dedicación y empeño para un futuro y sea una excelente profesional.

A mi mamá y mi papá, que siempre han estado en mi vida, por tanto amor y entrega, por ser ejemplo de sacrificio, por permitirme crecer…

A mi hermana que ha sido la raíz de mis proyecciones como profesional y convertirse para mí en un ejemplo a seguir…

A la memoria de mi abuela Ramona que, aunque hoy no esté, ha sido y seguirá siendo el motor de mi vivir…

AGRADECIMIENTOS

En primer lugar, agradecer eternamente a toda mi familia… en especial a mi abuela madre Ramona que ha sido y seguirá siendo mi ejemplo de sacrificio, dedicación, humildad y valentía, a pesar de no estar viva. A mi mamá que siempre ha estado presente para guiarme, apoyarme y darme consejos. A mi papá que nos ha enseñado que las cosas hay que ganárselas y sacrificarse en todo momento. A mi hermana Irina que ha sido mi ejemplo a seguir como profesional y como persona. Y a mi hija Carolina, que este enorme sacrificio realizado es por el bien de todos y pensando siempre en ella.

A mi tutora la Dra. Pilar, por soportar mis dudas, mis incertidumbres y ser tan valiente al tutorar la tesis en el momento que lo necesité; a la Dra. Helena, por su ayuda incondicional. A todo el colectivo de profesores del posgrado que de cada uno tomé lo mejor de ellos, por formar parte de mi vocación profesional posgraduada, por su dedicación y esmero a la hora de enseñar. A mis compañeros de clases por su compañerismo, amistad, sinceridad, apoyo y esfuerzo.

A todas las personas y familias que he podido conocer durante estos años de estudios en este país, por su preocupación hacia mí…

Un agradecimiento a la CONACYT por el apoyo como becario y a la UNAM, por permitirnos desde lejos venir a estudiar, dejarnos crecer como profesionales y dar hoy un “clic” al mundo de la computación.

A todos, ¡muchas gracias!

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

Contexto

El conocimiento es el tesoro más valioso de la raza humana. Gran parte del conocimiento existe en forma de lenguaje natural: libros, periódicos, informes técnicos, etcétera. En la actualidad se generan millones de datos a diario y gran parte de ellos son no estructurados, es decir, expresados en lenguaje natural que un humano puede comprender fácilmente, pero los softwares tradicionales no.

Hoy en día, existe una gran cantidad de investigadores en todo el mundo que exponen sus conocimientos a partir de la publicación de sus aportes en revistas científicas generados en formato de documentos; contribuyendo a la construcción colectiva del conocimiento, y a su vez a que otros investigadores avancen en un campo específico de investigación. El manejo cada vez más creciente de información en formatos no estructurados presenta como un problema para académicos a la hora de preguntarse cómo recopilar, explorar y aprovechar toda esta información, lo que hace necesaria la denominada minería de textos (del inglés, *text mining*)

La Universidad Nacional Autónoma de México (UNAM) cuenta con varias entidades que realizan proyectos de docencia, investigación y desarrollo tecnológico donde usan o aportan a las Ciencias de la Computación. La UNAM se ha distinguido en el ámbito mundial por sus aportaciones científicas y tecnológicas. Por otro lado, existen académicos que se han tardado en tiempo para obtener un resultado debido a la carencia de encontrar o vincularse con otros colegas de centros o institutos dentro de la propia universidad que estén realizando lo mismo o se puedan complementar y ayudar en ese sentido. Desafortunadamente, algunos campos producen demasiados artículos de muchas fuentes distintas, lo que dificulta la labor de analizar y clasificar tales documentos.

Planteamiento del problema

El problema consiste en el análisis de artículos científicos generados por académicos pertenecientes a diversos grupos de investigación o docencia o bien de forma aislada. Un académico puede escribir un artículo con dos colabores más y escribir diversos artículos con colaboradores diferentes cada vez. El académico no tiene la capacidad de

saber si el problema que está planteando resolver ya lo está atacando otro académico. Tampoco sabe a quién acudir si algún problema sale de su área de especialización y lo deja como trabajo futuro o lo resuelve desde cero.

El problema en términos del área de conocimiento es la identificación, comparación y mejora de algoritmos para el análisis de textos, provenientes de los artículos de investigación, docencia y desarrollo tecnológico en el área de conocimiento pertenecientes a la computación en la UNAM.

Con el propósito de solucionar el problema planteado se define como **objetivo general** de la investigación: Desarrollar nuevas estrategias en la ejecución de algoritmos de minería de textos para un mejor resultado en el rendimiento del procesamiento de artículos científicos en el área de las Ciencias de la Computación de la Universidad Nacional Autónoma de México.

Se desglosan a continuación los siguientes **Objetivos específicos:**

1. Formular el marco teórico de la investigación sobre los conceptos, etapas, herramientas, técnicas y métricas de evaluación empleadas para el análisis de textos.
2. Desarrollar nuevas estrategias en la ejecución de algoritmos de minería de textos con base al punto anterior, para un mejor resultado en su rendimiento.
3. Proponer una comparación del desempeño de algoritmos de minería de textos para el procesamiento de artículos científicos.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de la investigación:**

T1. Clasificar a partir de su contenido un conjunto de documentos que se incorporen por área de conocimiento.
T2. Clasificar a partir de su resumen (del inglés, *abstract*) un conjunto de documentos que se incorporen por área de conocimiento.
T3. Clasificar a partir de sus palabras claves (del inglés, *keywords*) un conjunto de documentos que se incorporen por área de conocimiento.
T4. Identificar cuáles palabras claves son comunes a lo largo de diferentes áreas de conocimiento.
T5. Identificar cuáles temas son compartidos por diferentes áreas de conocimiento.

Hipótesis

Se puede analizar el desempeño de algoritmos de minería de textos para un mejor resultado en el rendimiento del procesamiento en artículos científicos de proyectos de investigación y desarrollo de tecnología escritos por académicos de las Ciencias de la Computación de la UNAM.

Metodología

Esta propuesta de tesis va dirigida al análisis en el procesamiento de información en artículos científicos generados por académicos de la UNAM en el área de las Ciencias de la Computación, mediante algoritmos automatizados utilizados y desarrollados en la minería de textos. Tal objetivo se puede llevar a cabo a partir de la identificación de las diferentes etapas del procesamiento de documentos, y se amplía siguiendo las siguientes etapas:

1. Investigación documental para identificar los diferentes estudios y teorías relacionadas con el procesamiento automatizado de documentos.
2. Estudio de las características, técnicas, metodología y aplicación de la minería de textos en el procesamiento de documentos digitales.
3. Selección del corpus, a partir de la búsqueda de artículos científicos escritos por académicos de la UNAM de proyectos de investigación y desarrollo de tecnología, en revistas y eventos científicos como seminarios y coloquios.
4. Comprensión y estudio de la estructura conformada en artículos académicos.
5. Empleo de algoritmos de información documental basado en un método de tipo supervisado, conformado por dos etapas: aprendizaje y reconocimiento; además de otros tipos de algoritmos.
6. Análisis de la evaluación de los algoritmos empleados, así como la posible modificación de sus parámetros medidos para un mejor rendimiento del modelo.
7. Presentación, análisis de resultados y especificación de conclusiones para dar cumplimiento al objetivo propuesto.

Contribución y relevancia

Desde el punto de vista de aplicación a la UNAM:

El prototipo permitirá a los usuarios académicos realizar análisis de textos usando algoritmos de minería de textos para la búsqueda de artículos relevantes en su área de interés.

Desde el punto de vista científico:

Proponer nuevas estrategias en la ejecución de algoritmos de minería de textos, bajo determinadas circunstancias, en el rendimiento del procesamiento de diferentes artículos de proyectos de investigación y desarrollo de tecnología escritos por académicos de la UNAM.

La **estructura del trabajo** queda constituida de la siguiente manera:

Introducción: Planteamiento del contexto actual, el problema a resolver y el objetivo a alcanzar, así como la metodología a seguir para el logro de los objetivos específicos, las tareas y la contribución y relevancia propuestos.

Capítulo 1 Marco Teórico: Se explica una breve reseña sobre los conceptos principales referentes a la minería de textos, así como las herramientas, técnicas y métricas de evaluación para el procesamiento de textos, como fundamento teórico para el desarrollo y discusión de los resultados en función de la problemática del trabajo en cuestión.

Capítulo 2 Estado del Arte: Se realiza una breve explicación de algunos trabajos académicos realizados respecto al análisis de textos y documentos sirviendo de fundamento y base para la presente investigación.

Capítulo 3 Herramientas y tecnologías para el desarrollo de la propuesta de solución: Se establecen las bases para el desarrollo, en cuanto a las herramientas y tecnologías a emplearse, presentándose algunas de sus características fundamentales en función de la propuesta de solución.

Capítulo 4 Desarrollo de la propuesta de solución: Se detalla la propuesta de la arquitectura hacia la solución del problema tratado, y la implementación de los algoritmos de minería de textos a partir de la variación de sus parámetros y evaluación de rendimiento y aprendizaje; la consulta al corpus de documentos académicos buscados.

Capítulo 5 Presentación y discusión de resultados de la propuesta de solución: Se trata del análisis de resultados, cualitativos y cuantitativos de los algoritmos generados y, la evaluación a partir de las diferentes métricas; determinar cuáles son los mejores resultados obtenidos.

Conclusiones generales: Se llegan a conclusiones generales, dando cumplimiento de los objetivos trazados durante la investigación.

Trabajo a futuro: Se presentan otras alternativas para la continuidad en trabajos futuros en base a los resultados obtenidos y las observaciones llevadas a cabo.

Capítulo 1: Marco Teórico

Introducción

Hoy en día, la propagación del uso de dispositivos computacionales y de comunicación para la producción de información digital, y en particular en la producción de documentos textuales estructurados, semi - estructurados y no estructurados ha generado la necesidad de desarrollar métodos, algoritmos y sistemas capaces de realizar el procesamiento automatizado de éstos para la recopilación, exploración y aprovechamiento de toda la información, y con ello el surgimiento de áreas de estudio de la información como la minería de textos.

En el presente capítulo se describen los conceptos básicos referentes al procesamiento del lenguaje natural y la minería de textos, así como la relación de este último con otras disciplinas y las etapas para su desarrollo. Se fundamentan las técnicas más empleadas en el área, así como las métricas de evaluación en modelos predictivos de clasificación; sirviendo como fundamento teórico para el desarrollo y discusión de los resultados en función de la problemática del trabajo en cuestión.

1.1 Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (abreviado PLN, o NLP del inglés, *Natural Language Processing*) surge a partir de la interrelación que existe entre la lingüística, que es la ciencia que estudia el lenguaje humano, y la computación. A esta disciplina se le asignan diferentes nombres como: procesamiento de lenguaje natural, procesamiento de texto, tecnologías de lenguaje y lingüística computacional, en cualquiera de los casos se trata de procesar el texto por su sentido y no como un archivo binario.

El PLN, es un área encargada del desarrollo eficiente de algoritmos para procesar textos y hacer la información accesible a las aplicaciones informáticas. El alcance del procesamiento puede contener diferentes aplicaciones desde la identificación de palabras simples o compuestas hasta representaciones sintácticas jerárquicas de un alto nivel lógico de representaciones de colecciones de documentos. Las tareas más comunes son la segmentación de palabras, etiquetado de la oración – *Part of Speech*; POS -, desambiguación de palabras, análisis sintáctico, la alineación del texto, y extracción de frases (Tao, Zheng, Li, & Li, 2009).

Para la Asociación Mexicana de Procesamiento del Lenguaje Natural (AMPLN), las acciones principales del PLN son la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y computadores por medio del lenguaje natural, dicha comunicación es llevada a cabo por mecanismos eficaces computacionalmente, es decir, por programas de computadores que ejecuten o simulan la comunicación (AMPLN, 2019). El procesamiento del lenguaje natural sigue las siguientes etapas:

- Primero, el texto no se procesa directamente, sino se transforma en una representación formal que preserva sus características relevantes para la tarea o el método específico.
- Segundo, el programa principal manipula esta representación, transformándola según la tarea, busca en ella las subestructuras necesarias, etcétera.
- Finalmente, si es necesario, los cambios hechos a la representación formal – o la respuesta generada en esta forma – se transforma en lenguaje natural.

1.2 Minería de textos y su relación con otras áreas

Partiendo de lo planteado anteriormente, surge la minería de textos como un enfoque particular del proceso de descubrimiento de conocimiento, específicamente, orientado al descubrimiento en fuentes textuales y no estructuradas. A partir del estudio realizado en las bibliografías consultadas y para un mejor entendimiento del concepto de minería de textos, se exponen a continuación algunas definiciones planteadas por diferentes autores sobre esta temática.

La minería de textos se define como el proceso de descubrimiento de patrones interesantes y nuevos conocimientos en una colección de textos, es decir, es el proceso encargado del descubrimiento de conocimientos que no existían explícitamente en ningún texto de la colección, pero que surgen de relacionar el contenido de varios de ellos (Hearst, 1999) (Kodratoff, 1999).

Descubrimiento de información útil y previamente desconocida a partir de texto sin estructurar (Xu, Kurz, & Piskorski, 2002).

La minería de textos (Weiss, Indurkhya, Zhang, & Damerau, 2005) cuenta con un gran número de herramientas para la extracción automática de información, cada

herramienta trabaja con el texto en distintos niveles de complejidad. De manera general se pueden enumerar los siguientes:

- Léxico: Trata a las palabras y términos como elementos individuales independientes de su contexto, enfocándose en sus propiedades numéricas y estadísticas.
- Sintáctico: Agrupa palabras y términos en oraciones o fragmentos de oraciones, donde cada uno desempeña una función dentro de la misma oración.
- Semántico: Intenta determinar el significado de las oraciones o de un conjunto de oraciones a través de un análisis profundo del texto.

Para (Sánchez & Martín-Bautista, 2008) la minería de textos hace referencia al descubrimiento no trivial potencialmente útil de conocimiento partiendo de una colección de documentos de texto no estructurado. Y puede realizarse una analogía con la minería de datos, encargada de descubrir conocimiento en bases de datos.

Según las consultas realizadas en esta investigación basadas en las definiciones de la literatura científica, se asume como minería de textos al proceso de identificar y descubrir nuevo conocimiento a partir de un conjunto de documentos textuales sin estructurar.

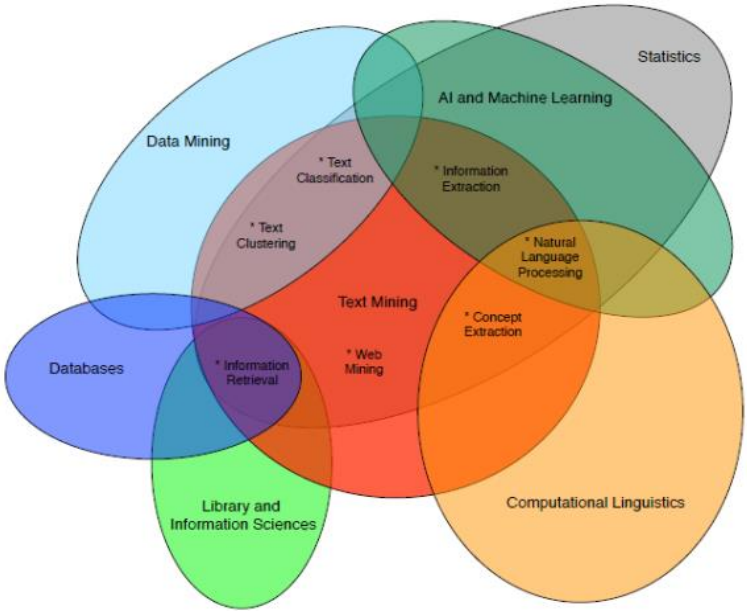

Fig. 1 Áreas relacionadas de la minería de datos

Fuente: Miner, G., Elder, J., & Nisbet, B. (2012). Practical text mining and statistical analysis for non-structured text data applications. Waltham, Massachusetts: Academic Press.

La minería de textos utiliza diversos métodos y tecnologías para el análisis de los documentos desarrollados por otras áreas para el logro de sus objetivos. En la figura (Fig. 1) se muestran las áreas relacionadas con la minería de textos, como lo son las bases de datos, la minería de datos, la estadística, el aprendizaje de máquina, el procesamiento del lenguaje natural, la recuperación de la información, la extracción de la información, entre otras. A continuación, se explica de manera breve la clara relación existente entre minería de textos y minería de datos, recuperación de información y lingüística computacional.

1.2.1 Minería de textos y minería de datos

La combinación de minería de textos y minería de datos ofrece un punto de vista más amplio que el de los datos estructurados o el de los datos no estructurados por separado. A veces, la minería de textos resulta una actividad complementaria de la minería de datos. Esta última es definida como “el proceso que tiene como propósito descubrir, extraer y almacenar información relevante de amplias bases de datos, a través de programas de búsqueda e identificación de patrones y relaciones globales, tendencias, desviaciones y otros indicadores aparentemente caóticos que tienen una explicación que pueden descubrirse mediante diversas técnicas de esta herramienta” (Ángeles L. & Santillán G., 2001).

La minería de datos tiene una gran variedad de ámbitos de aplicación, siendo prácticamente útil en todas las facetas de la actividad humana; algunas de las tareas más importantes son las siguientes:

- Comercio y banca: segmentación de clientes, previsión de ventas, análisis de riesgo.
- Medicina y farmacia: diagnóstico de enfermedades y la efectividad de los tratamientos.
- Seguridad y detección de fraude: reconocimiento facial, identificaciones biométricas, acceso a redes, etcétera.
- Recuperación de información no numérica: minería de textos, minería web, búsqueda e identificación de imagen, video, voz y texto de bases de datos multimedia.
- Astronomía: identificación de nuevas estrellas y galaxias.

- Geología, minería, agricultura y pesca: identificación de áreas de uso para distintos cultivos o de pesca o de exploración en bases de datos de imágenes de satélites.
- Ciencias ambientales: identificación de modelos de funcionamiento de ecosistemas naturales y/o artificiales (plantas depuradoras de aguas residuales) para mejorar su observación, gestión y/o control.
- Ciencias sociales: estudios de los flujos de la opinión pública.
- Planificación de ciudades: identificar barrios con conflicto en función de valores sociodemográficos (Riquelme, Ruíz, & Gilbert, 2006).

A partir de ellos, se puede evidenciar que la minería de datos es una herramienta muy importante para poder interpretar, por ejemplo, los rumbos de una empresa tomando en cuenta datos históricos obtenidos a través del tiempo, este tipo de minería podrá descubrir tendencias que existan acerca de algún problema relacionado con la organización o podrá otorgarle algún tipo de ventaja con el conocimiento de cierta información exclusiva, descubierta a través de la utilización del ciclo de la minería de datos. Por su parte, la minería de textos aporta casi lo mismo que la minería de datos, pero con el valor añadido del descubrimiento de nuevos conocimientos partiendo de un gran conjunto de textos.

1.2.2 Minería de textos y la recuperación de información

La recuperación de información (RI) es el conjunto de actividades orientadas a facilitar la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros. El esquema general de un sistema de RI se puede visualizar en la figura (Fig. 2). Un concepto básico para la RI es medir la similitud entre documentos. Generalmente se realiza entre dos de ellos, pero también se pude considerar, para fines de búsqueda, al conjunto de palabras ingresadas por el usuario como un documento. Las principales medidas de similitud son: comparación semántica (cantidad de palabras que comparten, palabras de diccionario que comparten) y comparaciones matemáticas (coeficiente cosenoidal) (Gösling & J., 2009).

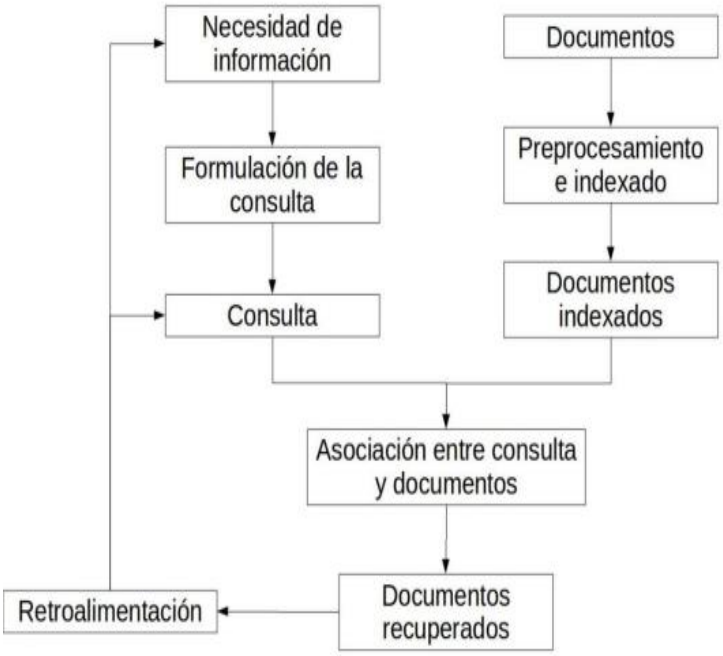

Fig. 2 Esquema de un sistema de recuperación de información

Fuente: Hiemstra, D. (2009). Information retrieval models. Information Retrieval: searching in the 21st Century, 2-19.

Sin embargo, no pretende la RI, facilitar el proceso de extracción y análisis de nuevos conocimientos, al contrario que la minería de textos. Además, cabe tener en cuenta que el objetivo de la RI se centra en tomar únicamente documentos útiles que satisfagan las necesidades del usuario mientras que la minería de textos no solo no tiene esta necesidad, sino que necesita de una pregunta concreta para realizar el estudio.

1.2.3 Minería de textos y la lingüística computacional

La lingüística computacional o lingüística informática (del inglés, *computacional linguistics*) es un campo científico interdisciplinar relativamente reciente, cuyo objetivo radica en incorporar en las computadoras la habilidad en el manejo del lenguaje humano (Gómez G., 1998). Desde el punto de vista de su vinculación a la informática, suele ser considerada como una subdisciplina de la Inteligencia Artificial (IA). Por otra parte, desde el punto de vista de su vinculación a la lingüística, puede ser considerada una subdisciplina de la lingüística teórica. Por último, en cuanto que disciplina experimental orientada a la elaboración de productos comerciales y de investigación, forma parte de las denominadas industrias de la lengua, un sector industrial cada vez más amplio que proporciona datos y programas informáticos aplicados al tratamiento del lenguaje.

La finalidad de la lingüística computacional es el estudio gramatical y sintáctico de los documentos en formato electrónico, usando técnicas para procesarlos con el fin de que puedan ser compresibles para un ordenador. Si bien la minería de textos necesita de la lingüística computacional, no comparten objetivo.

1.3 Etapas de la minería de textos

La metodología empleada para realizar la minería de textos puede ser general o específica. Una metodología general como la propuesta por (Tan, 1999), en el que define dos etapas principales: una etapa de pre - procesamiento y una etapa de descubrimiento. En la primera etapa, los textos se transforman a algún tipo de representación estructurada o semi - estructurada que facilite su posterior análisis, mientras que en la segunda etapa las representaciones intermedias se analizan con el objetivo de descubrir en ellas algunos patrones interesantes o nuevos conocimientos. La figura (Fig. 3) ilustra este proceso.

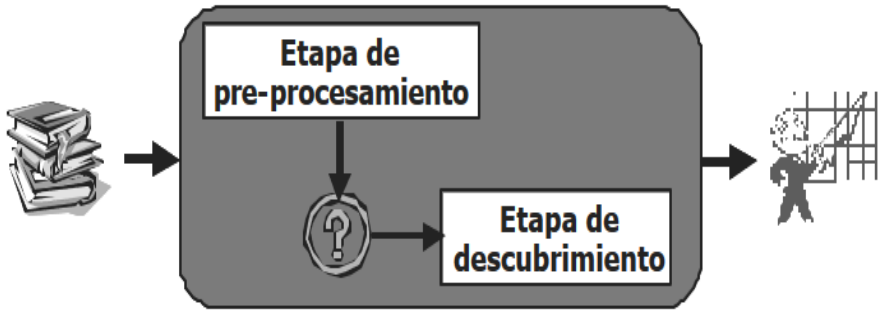

Fig. 3 Proceso de minería de textos

Fuente: Montes y Gómez, M. (2001). Minería de texto: Un nuevo reto computacional. Obtenido de https://ccc.inaoep.mx/~mmontesg/publicaciones/2001/MineriaTexto-md01.pdf

Una metodología más específica propuesta por (Sukanya M., 2012), establece una serie de pasos o etapas descritas a continuación:

- El primer paso es determinar el propósito de estudio de la minería de textos, por ejemplo, en el caso de textos de biología se puede identificar y etiquetar entidades biológicas, extracción y normalización de sinónimos, homónimos y abreviaturas, identificación de entidades biológicas, etcétera.
- El segundo paso es recolectar, identificar, recoger y validar información. En esta fase se realiza la RI en la cual se buscan e identifican las fuentes más relevantes para el objetivo de estudio de la minería de textos. Luego, se deben recopilar los

documentos detectados en el mejor formato, se seleccionan, se evalúa su relevancia y se realizan las anotaciones necesarias. Aquí se cuenta con el conjunto de documentos necesarios para la realización de la minería de textos.

- En el tercer paso se realiza el procesamiento de texto con la finalidad de eliminar información que no ayuda al propósito de la minería de textos, realizándose algunas de las siguientes acciones: análisis léxico, tratamiento y separación de palabras vacías (artículos, preposiciones, conjunciones), tratamiento de términos flexionados (términos relacionados morfológicamente, variaciones de género, número o tiempo verbal), tratamiento de palabras compuestas, normalización de palabras, obtención de las raíces de las palabras y etiquetado de palabras, además de corregir algunos problemas que presenten los documentos como: los problemas de formato, polisemia, homonimia, sinonimia. Esta fase exploratoria se basa principalmente en lingüística computacional (análisis morfológico y sintáctico), además de algoritmos informáticos (Abbott, 2013). Su objetivo es facilitar la selección de características deseadas para identificar palabras clave, identificación de entidades biológicas, individuos, organizaciones, lugares, oraciones, conceptos, etcétera.
- En el cuarto paso se realiza la extracción y análisis de clases, relaciones, asociaciones o secuencias, con el fin de encontrar evidencias de conceptos y de estructuras existentes. Durante esta etapa los documentos se pueden representar a través del modelo del espacio vectorial (Salton, 1989), donde cada documento es modelado como un vector de dimensión n y es representado de la forma $D_i = (d_{i1}, d_{i2}, \dots, d_{ij})$ donde cada d_{ij} representa el número de repeticiones de términos en el documento. Los datos obtenidos en esta etapa son representados en alguna estructura informática que facilita su análisis; las estructuras representan las relaciones entre las entidades de un mismo tipo de datos, palabras o conceptos clave, documento - términos, términos - autores, etcétera.
- En el último paso se presentan los resultados para su interpretación mediante resúmenes, marcados de texto, relaciones, taxonomías y visualización. En esta etapa también se puede almacenar la información procesada en bases de datos para su recuperación posterior.

De manera resumida la figura (Fig. 4) muestra la metodología de la minería de textos, antes explicada.

Fig. 4 Metodología de la minería de textos.

Fuente: Contreras Barrera, M. (2016). Minería de texto en la clasificación de material bibliográfico. Obtenido de https://www.redalyc.org/jatsRepo/161/16148511003/html/index.html

1.4 Tareas para el procesamiento de textos

Cuando se enfrenta al texto con la idea de descubrir conocimiento, se encuentra con el problema de la falta de estructura del mismo. Esta falta de estructura es solo aparente, porque, realmente, el texto presenta una estructura demasiado compleja y difícil de tratar computacionalmente. Dependiendo del tipo de operaciones usadas en este pre – procesamiento de datos, será el tipo de patrones a descubrir en esta colección. A continuación, se describen algunas de las tareas para el pre - procesamiento con la finalidad de categorizar automáticamente documentos de textos, y que posteriormente serán empleadas para el desarrollo de la propuesta de solución.

1.4.1 Tokenizador

La tokenización es la forma de separar el texto en palabras comúnmente llamadas tokens, usando algunos caracteres como referencia para dividir. Este proceso toma en cuenta que las palabras pueden estar interrumpidas por un final de línea, están pegadas a signos de puntuación, no siempre están separadas por espacios y no siempre los espacios en blanco separan las palabras. Los tokens son generalmente palabras y símbolos de puntuación. La tokenización no implica ningún nivel de análisis y se realiza muy rápido.

1.4.2 Stemmer

Los algoritmos de *stemming* intentan reducir las palabras flexionadas y derivadas en su forma raíz, es decir, extraer la raíz de una palabra, la raíz lingüística a la que pertenece; generalmente son algoritmos muy rápidos. El *stemming* es una forma de análisis morfológico. Este proceso se realiza porque la raíz de una palabra puede aparecer más veces en un texto. El algoritmo más común para *stemming* es el algoritmo de Porter. Existen además métodos basados en análisis lexicográfico y otros algoritmos similares (KSTEM, *stemming* con cuerpo, métodos lingüísticos, entre otros).

El algoritmo de Porter (PorterStemmer, 2006) es un algoritmo de derivación de palabras y consiste en eliminar las terminaciones morfológicas e inflexionales comunes de las palabras en inglés. Sin embargo, existen otras modificaciones de dicho algoritmo, el algoritmo de Porter2 (Snowball, 2006), para este algoritmo es mejor si se usan textos en español. Para este último, se hace uso del *Snowball*, un pequeño lenguaje de programación para el manejo de *strings* que permite implementar fácilmente algoritmos de *stemming*.

1.4.3 Etiquetador gramatical

El etiquetado de parte del discurso (del inglés, *part-of-speech*, POS) es el proceso de marcar una palabra en un texto con una parte particular del discurso, en función de su definición y contexto. Captan información sintáctica y son útiles, por ejemplo, para identificar las intenciones del usuario en los tweets (Gomez-Adorno, Bel-Enguix, Sierra, Sánchez, & Quezada, 2018). Los etiquetadores POS generalmente requieren un corpus marcado manualmente. El etiquetado POS es una forma de análisis morfo-sintáctico. Existen una variedad de bibliotecas y servicios que ayudan a obtener el etiquetado POS de un documento; las cuales en el próximo capítulo será explicado.

1.4.4 Lematizador

Lematización de los términos, es una parte del procesamiento lingüístico que trata de determinar el lema de cada palabra que aparece en un texto. Su objetivo es reducir una palabra a su raíz, de modo que las palabras clave de una consulta o documento se representen por sus raíces en lugar de por las palabras originales. El lema de una palabra comprende su forma básica más sus formas declinadas. La lematización requiere etiquetado POS. La lematización requiere un diccionario como *WordNet* o

Wikipedia. La lematización es un proceso computacionalmente costoso, en comparación con el *stemming*.

1.5 Técnicas en minería de textos

La minería de textos junto a las técnicas de RI, Extracción de Información, *Machine Learning* (del español, Aprendizaje Automático) o IA, ha ampliado su ámbito de aplicación como, en aplicaciones para web semántica, redes sociales o herramientas de *help-desk*. Por lo que se realiza una revisión en la literatura científica referente a las técnicas más utilizadas en el área, lo cual sirvió como base de implementación en el desarrollo de la propuesta de solución.

1.5.1 Aprendizaje supervisado

En el aprendizaje supervisado, los algoritmos trabajan con datos "etiquetados" (del inglés, *labeled data*), para deducir una función a partir de datos de entrenamiento que, dadas las variables de entrada (del inglés, *input data*), les asigne la etiqueta de salida adecuada. Se entrena un modelo con ejemplos predefinidos y luego se usa ese modelo para predecir los resultados de los datos de entrada.

Estos dos tipos principales de aprendizaje supervisado: clasificación y regresión, se distinguen por el tipo de variable objetivo. En los casos de clasificación, es de tipo categórico, mientras que, en los casos de regresión, la variable objetivo es de tipo numérico. Se explica a continuación, algunos de los algoritmos más frecuentes de aprendizaje supervisado aplicados al área de la minería de textos, que funcionan tanto para problemas de clasificación y regresión, tanto binario y/o multi – clase, siendo ésta última, objeto de estudio de la presente investigación.

2.6.1.1 Naïve Bayes y Multinomial Naïve Bayes

Los métodos de Naïve Bayes son un conjunto de algoritmos de aprendizaje supervisado basados en la aplicación del teorema de Bayes con la suposición de independencia "ingenua" entre cada par de características. Se le llaman "ingenuo" porque asumen que las características son independientes dada la clase. El clasificador Naïve Bayes es un clasificador generativo que modela la distribución de documentos en cada clase, utilizando un modelo probabilístico con supuestos independientes sobre las distribuciones de diferentes términos. Para (Aggarwal & Zhai, 2012) los clasificadores probabilísticos fueron diseñados para usar un modelo implícito de mezcla para generar

documentos subyacentes. Cada componente de la mezcla es un modelo generativo que proporciona la probabilidad de un término dado con ejemplos para dicho componente.

Naïve Bayes es uno de los algoritmos de aprendizaje inductivo más eficientes y eficaces para el aprendizaje automático y la minería de datos (Zhang, 2004). A pesar de ser fácil de comprender y funcionar bien con datos de alta dimensión. Este algoritmo tiene un sorprendente desempeño competitivo en la clasificación, porque la suposición de independencia condicional en la que se basa, rara vez es cierto en las aplicaciones del mundo real, es decir, la suposición de que las características son condicionalmente independientes dada la clase no es realista.

Dada una variable de clase y y un vector de características dependientes x_1 a x_n, el teorema de Bayes establece la siguiente relación (1) (scikit-learn, 1.9. Naive Bayes, s.f.):

$$P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)} \quad (1)$$

De esta manera, usando el supuesto independencia de ingenuo, quedaría (2):

$$P(x_i\ |y, x_1, \dots, x_{i-1}, x_{i+1}, \dots\ , x_n) = P(x_i|y) \quad (2)$$

para toda i, esta relación es simplificada a (3):

$$P(y|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots, x_n)} \quad (3)$$

Como $P(x_1, \dots, x_n)$ es una constante debido a la entrada, se puede usar la siguiente regla de clasificación (4):

$$P(y|x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \ \Rightarrow\ \hat{y} = \arg\ \max_y P(y)\prod_{i=1}^{n} P(x_i|y) \ (4)$$

de esa manera se puede usar la estimación Máxima A Posteriori (MAP) para estimar a $P(y)$ y $P(x_i|y)$; el primero es entonces la frecuencia relativa de la clase y en el conjunto de entrenamiento.

Entre la amplia variedad de clasificadores existentes del método de Bayes, se puede citar: *Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes y Out-of*

core Naïve Bayes model fitting. Estos clasificadores difieren principalmente por las suposiciones que hacen respecto a la distribución de $P(x_i|y)$.

2.6.1.2 Árboles de decisión

Los árboles de decisión (del inglés, *Decision Tree,* DT) son un método de aprendizaje supervisado no paramétrico utilizado para la clasificación y la regresión. El objetivo es crear un modelo que prediga el valor de una variable objetivo mediante el aprendizaje de reglas simples de decisión inferidas a partir de las características de los datos. Utiliza un DT como modelo predictivo que mapea observaciones sobre un artículo a conclusiones sobre el valor objetivo del artículo. Las hojas representan etiquetas de clase y las ramas las características que conducen a esa clase. Cada nodo representa uno de los valores de entrada y sus nodos hijos representan un posible valor para esa variable de entrada. Las hojas representan un posible valor para la entrada dada (scikit-learn, 1.10. Decision Trees, s.f.).

En el ejemplo (Fig. 5), los DT aprenden de los datos para aproximar una curva sinusoidal con un conjunto de reglas de decisión *if-then-else*; cuanto más profundo es el árbol, más compleja es la decisión y más ajustado es el modelo.

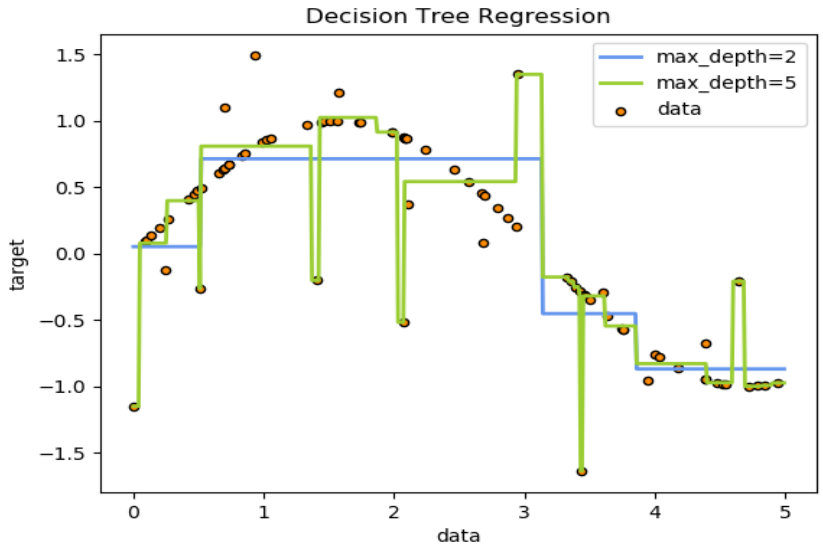

Fig. 5 Decision Tree Regression

Fuente: (scikit-learn, 1.10. Decision Trees, s.f.)

Algunas de las ventajas de los DT son:

- Simple de entender e interpretar; los árboles pueden ser visualizados.

- Requiere poca preparación de datos. Otras técnicas a menudo requieren normalización de datos, se deben crear variables ficticias y se deben eliminar los valores en blanco.
- El costo de usar el árbol (es decir, predecir datos) es logarítmico en la cantidad de puntos de datos utilizados para entrenar el árbol.
- Capaz de manejar problemas de salida múltiple.
- Utiliza un modelo de caja blanca. Si una situación dada es observable en un modelo, la explicación de la condición se explica fácilmente por la lógica booleana. Por el contrario, en un modelo de caja negra (por ejemplo, en una red neuronal artificial), los resultados pueden ser más difíciles de interpretar.
- Posibilidad de validar un modelo usando pruebas estadísticas. Eso hace posible dar cuenta de la fiabilidad del modelo.

Las desventajas de los DT incluyen:

- Los DT pueden ser inestables porque pequeñas variaciones en los datos pueden dar como resultado la generación de un árbol completamente diferente. Este problema se mitiga usando DT dentro de un conjunto.
- Se sabe que el problema de aprender un DT óptimo es NP-completo en varios aspectos de optimalidad e incluso para conceptos simples. En consecuencia, los algoritmos prácticos de aprendizaje del DT se basan en algoritmos heurísticos como el algoritmo *greedy* donde se toman decisiones localmente óptimas en cada nodo. Tales algoritmos no pueden garantizar devolver el DT óptimo globalmente.
- Hay conceptos que son difíciles de aprender porque los DT no los expresan fácilmente, como los problemas de XOR, paridad o multiplexor.
- En caso de dominarse algunas clases, se utilizan árboles sesgados, recomendándose equilibrar el conjunto de datos antes de ajustarlo al DT.

2.6.1.3 Regresión Logística

La regresión logística (del inglés, *logistic regression*), a pesar de su nombre, es un modelo lineal para la clasificación en lugar de la regresión (scikit-learn, 1.1.11. Logistic regression, s.f.). También se conoce en la literatura como *logit regression*, *maximum-entropy classification* (MaxEnt) (del español, clasificación de máxima entropía) o el *log-linear classifier* (del español, clasificador log-lineal).

Se utiliza para predecir la probabilidad de una variable dependiente categórica. En la regresión logística, la variable dependiente es una variable binaria que contiene datos codificados como 1 – 0, sí – no, abierto – cerrado, etcétera. Es útil para modelar la probabilidad de un evento ocurriendo como función de otros factores. El análisis de regresión logística se enmarca en el conjunto de Modelos Lineales Generalizados (GLM por sus siglas en inglés) que usa como función de enlace la función *logit*. Las probabilidades que describen el posible resultado de un único ensayo se modelan, como una función de variables explicativas, utilizando una función logística (5).

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \quad (5)$$

Donde:

- $e = base\ del\ logaritmo\ natural\ (constante\ de\ Euler)$
- $x_0 = valor\ del\ punto\ medio\ del\ sigmoide$
- $L = máximo\ valor\ de\ la\ curva$
- $k = radio\ de\ crecimiento\ logístico\ o\ inclinación\ de\ la\ curva$

El modelo de regresión logística binaria tiene extensiones a más de dos niveles de la variable dependiente: las salidas categóricas con más de dos valores se modelan mediante regresión logística multinomial. En estadística, la regresión logística multinomial generaliza el método de regresión logística para problemas multiclase, es decir, con más de dos posibles resultados discretos (Greene, 2012). Se trata de un modelo que se utiliza para predecir las probabilidades de los diferentes resultados posibles de una distribución categórica como variable dependiente, dado un conjunto de variables independientes (que pueden ser de valor real, valor binario, categórico - valorado, etcétera).

La regresión logística requiere tamaños de muestra bastante grandes. La razón por la cual es ampliamente utilizada, a pesar de los algoritmos avanzados como redes neuronales profunda, es porque es muy eficiente y no requiere demasiados recursos computacionales que hacen que sea asequibles ejecutar la producción.

2.6.1.4 Máquinas de vectores de soporte

Las máquinas de vectores de soporte (SVM) son un conjunto de métodos de aprendizaje supervisados que se utilizan para la detección de clasificación, regresión y valores

atípicos. Dado un conjunto de ejemplos de entrenamiento, una SVM los representa como puntos en el espacio, separando las clases por la mayor distancia posible (Fig. 6). Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, según su proximidad serán asignadas a una u otra clase, es decir, si un punto nuevo pertenece a una categoría o la otra (scikit-learn, 1.4. Support Vector Machines, s.f.).

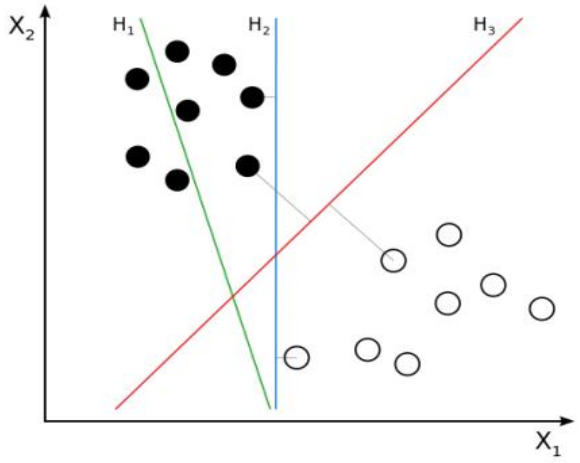

Fig. 6 H1 no separa las clases. H2 sí, pero H3 lo hace con la separación máxima

Fuente: https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

Las ventajas de las máquinas de vectores de soporte son:

- Son efectivos en casos donde el número de dimensiones es mayor que el número de muestras.
- Utilizan un subconjunto de puntos de entrenamiento en la función de decisión, llamados vectores de soporte, por lo que también es eficiente desde el punto de vista de la memoria.
- Son versátil, se pueden especificar diferentes funciones del *Kernel*[1] para la función de decisión. Se proporcionan núcleos comunes o se pueden definir núcleos personalizados para tipos de datos específicos.

Las desventajas de las máquinas de vectores de soporte incluyen:

- En caso que el número de funciones es mucho mayor que el número de muestras, se debe de evitar el ajuste excesivo al elegir las funciones del *Kernel* y el término de regularización es crucial.

[1] Un kernel es una medida de similitud (producto punto modificado) entre puntos de datos.

- Las SVM no proporcionan estimaciones de probabilidad directamente, estas se calculan utilizando una costosa validación cruzada de cinco veces.

Para el empleo en el desarrollo de dicha técnica es importante esclarecer el parámetro *C* (fuerza de la regularización[2]), teniendo las siguientes las características:

- Valores más grandes de C (menos regularización): significa que ajusta los datos de entrenamiento lo mejor posible y cada punto de datos individual es importante para clasificar correctamente.
- Valores más pequeños de C (más regularización): significa más tolerante a errores en puntos de dato individuales.

Para problemas de clasificación multi – clase (del inglés, *multi - class*) en un conjunto de datos como parte del conjunto de métodos de las SVM, se pueden citar los algoritmos: SVC (*C-Support Vector Classification*), NuSVC (*Nu-Support Vector Classification*) y LinearSVC (*Linear Support Vector Classification*). SVC y NuSVC implementan el enfoque de "*one-against-one*" ("uno contra uno") (Knerr, Personnaz, & Dreyfus, 1990) para la clasificación de clases múltiples. Por otro lado, LinearSVC implementa la estrategia multi - clase "*one-vs-the-rest*" ("uno frente al resto"), por lo que entrena a los modelos *n_class* (si solo hay dos clases, solo se entrena un modelo).

Para los métodos de SVM con *kernel*, la eficiencia (velocidad de tiempo de ejecución y uso de la memoria) disminuye a medida que aumenta el tamaño del conjunto de entrenamiento; y se necesita normalización de los datos de entrada y ajuste de parámetros. En este sentido, el LinearSVC su parámetro de *kernel* es lineal, teniendo como ventajas: simple y fácil de entrenar, predicción rápida, escala bien a conjuntos de datos muy grandes, funciona bien con datos dispersos y las predicciones son relativamente fáciles de interpretar. Sin embargo, para conjuntos de datos pequeños, otros modelos pueden tener un rendimiento de generalización superior.

1.5.2 Aprendizaje no supervisado

A diferencia del aprendizaje supervisado, en el aprendizaje no supervisado solo se le otorgan las características, sin proporcionarle al algoritmo ninguna etiqueta (no se

[2] Se refiere a regularización, a la capacidad de un algoritmo para proporcionar predicciones precisas para datos nuevos que no se habían visto anteriormente.

dispone de datos "etiquetados" para el entrenamiento); solo se conoce los datos de entrada, pero no existen datos de salida que correspondan a una determinada entrada.

Su función es la agrupación, por lo que el algoritmo debería catalogar por similitud y poder crear grupos, sin tener la capacidad de definir cómo es cada individualidad de cada uno de los integrantes del grupo. Así, el aprendizaje no supervisado típicamente trata los objetos de entrada como un conjunto de variables aleatorias, siendo construido un modelo de densidad para el conjunto de datos. Por ello, tienen un carácter exploratorio.

Se explica a continuación, algunos de los algoritmos más frecuentes de aprendizaje no supervisado aplicados al área de la minería de textos.

2.6.2.1 Agrupamiento

El c*lustering* (o algoritmo de agrupamiento) es una técnica de la minería de textos, que consiste en agrupar una serie de vectores según un criterio en grupos o *clusters*, generalmente el criterio suele ser la similitud entre ellos (scikit-learn, 2.3. Clustering, s.f.). Es un método no supervisado que realiza descubrimiento en agrupaciones de documentos (Fig. 7), basándose en su contenido o en un tema determinado que caracteriza a cada uno de los grupos en los que se puede dividir una colección de documentos (Justicia de la T., 2017). Para pre - procesar los documentos se puede combinar con formas intermedias como *bags of words*, o incluso se pueden enriquecer los documentos mediante ontologías (Zhao & Karypis, 2005).

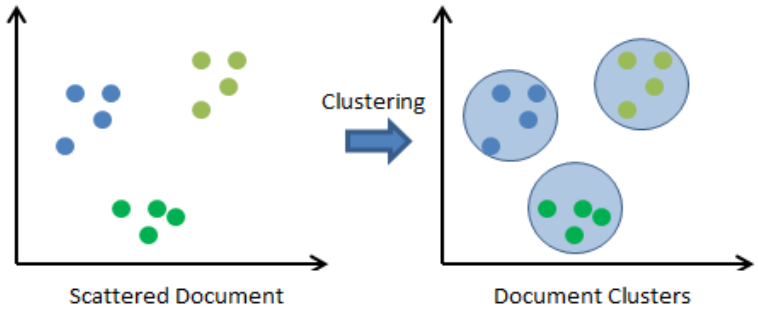

Fig. 7 Clustering de documentos

2.6.2.2 K-medias

El algoritmo de *K-means* (del español, K-medias) agrupa los datos tratando de separar las muestras en n grupos de igual varianza, minimizando un criterio conocido como

inercia o suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de clústeres, el cual particiona los N objetos en K particiones, en donde un objeto irá al clúster con la media μ_j más cercana. Los medios se denominan comúnmente "centroides" del grupo. El algoritmo se adapta bien a una gran cantidad de muestras y tiene como objetivo elegir centroides que minimicen la inercia, o la suma dentro del clúster del criterio de cuadrado (6) (scikit-learn, 2.3. Clustering, s.f.):

$$\sum_{i=0}^{n} \min_{\mu_j \epsilon C}(||x_j - \mu_j||^2) \quad (6)$$

K-means a menudo se conoce como el algoritmo de Lloyd. En términos básicos, el algoritmo tiene tres pasos. El primer paso elige los centroides iniciales, con el método más básico es elegir k muestras del conjunto de datos X. Después de la inicialización, consiste en un ciclo repetitivo entre los otros dos pasos. El primer paso asigna cada muestra a su centroide más cercano. El segundo paso crea nuevos centroides al tomar el valor medio de todas las muestras asignadas a cada centroide anterior. La diferencia entre los centroides antiguos y nuevos se calcula y el algoritmo repite estos dos últimos pasos hasta que este valor sea menor que un umbral. En otras palabras, se repite hasta que los centroides no se mueven significativamente.

2.6.2.3 Modelado de tópicos: Latent Dirichlet Allocation

El modelado de tópicos (del inglés, *topic modelling*), básicamente consiste en identificar tópicos o temas en textos, es decir, realizar un análisis de lo que hay en una colección de texto. Para la presente investigación se hace útil para detectar tendencias - descubrir temas compartidos - en los documentos por las diferentes áreas de conocimiento.

Para ello, se asume que un documento es una mezcla de temas y, por otra parte, los temas se representan como una distribución de palabras. Un tópico en el contexto de modelado de tópicos es una distribución de probabilidades de palabras para un conjunto, e indica la probabilidad que una palabra aparezca en un documento sobre un tópico en particular. Existen diferentes enfoques de modelado de temas: Análisis semántico latente probabilístico (PLSA), *Latent Dirichlet Allocation* (LDA) y Basada en aprendizaje profundo (LDA2VEC).

El modelado de *Latent Dirichlet Allocation* (Blei, Ng, & Jordan, 2003) asume que los documentos se producen a partir de una mezcla de temas. Esos temas luego generan

palabras basadas en su distribución de probabilidad. Dado un conjunto de datos de documentos, LDA realiza un seguimiento e intenta averiguar qué temas crearían esos documentos en primer lugar.

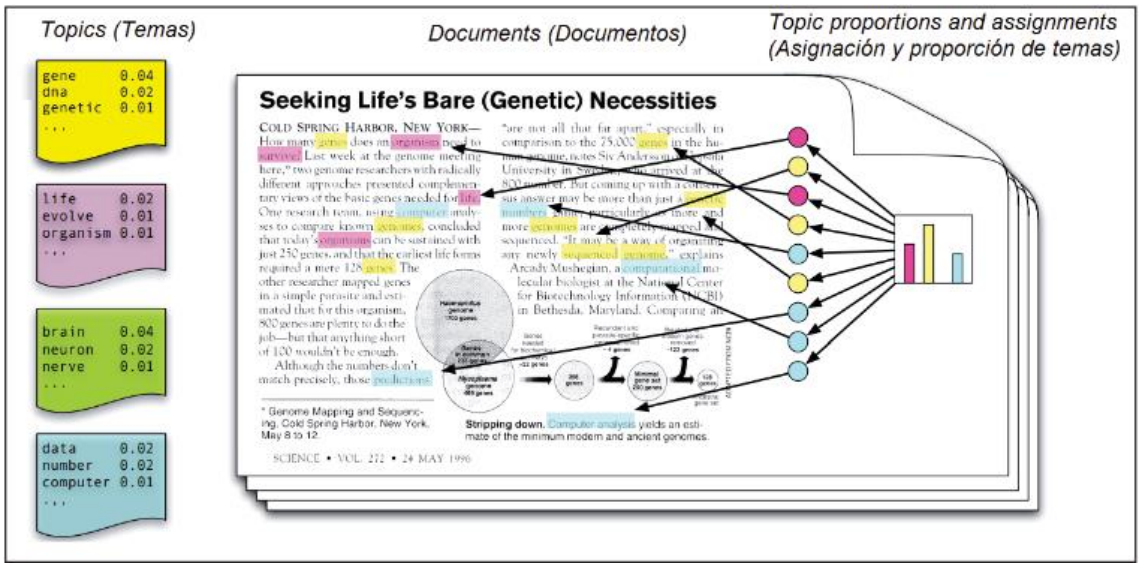

Fig. 8 La intuición detrás de la técnica LDA

Fuente: http://opac.pucv.cl/pucv_txt/txt-5000/UCD5100_01.pdf

En la figura (Fig. 8) se muestra un documento donde se pueden separar y distinguir las palabras que tienen que ver con análisis como "*computer*" y "*prediction*", las palabras que tienen que ver con biología evolutiva como "*life*" y "*organism*" y las palabras que hablan sobre genética como "*sequenced*" y "*genes*". Si se tomara más tiempo y se resaltara todas las palabras que tienen que ver con análisis, biología evolutiva y genética, se daría cuenta que todo el documento posee estos tres temas mezclados, pero en diferentes proporciones.

La técnica LDA es un modelo estadístico para colección de documentos que intenta capturar esta intuición. En esta se define un tema para pasar a ser una distribución sobre un diccionario de palabras fijado. Por ejemplo, en el documento analizado en la figura (Fig. 8) el tema "genética" tiene un vocabulario de palabras que poseen una alta probabilidad de pertenecer al tema "genética".

LDA es una técnica de factorización matricial (Fig. 9); convierte la Matriz de Término del Documento (Fig. 9a) en dos matrices de dimensiones inferiores: $M1$ y $M2$, utilizando técnicas de muestreo para mejorar estas matrices.

- $M1$ es una matriz de temas de documentos (Fig. 9b).
- $M2$ es una matriz de temas con dimensiones (N, K) y (K, M) respectivamente (Fig. 9c), donde N es el número de documentos, K es el número de temas y M es el tamaño del vocabulario.

	w_1	w_2	w_3	w_m
D_1	0	2	1	3
D_2	1	4	0	0
D_3	0	2	3	1
D_n	1	1	3	0

a. Matriz de Término Documento

	K_1	K_2	K_3	K_n
D_1	1	0	0	1
D_2	1	1	0	0
D_3	1	0	0	1
D_n	1	0	1	0

b. M1: Matriz de temas de documentos

	w_1	w_2	w_3	w_m
K_1	0	1	1	1
K_2	1	1	1	0
K_3	1	0	0	1
K_n	1	1	0	0

c. M2: Matriz de temas con dimensiones

Fig. 9 Factorización matricial del LDA

Itera a través de cada palabra "w" para cada documento "D" e intenta ajustar el tema actual. Se asigna un nuevo tema "K" a la palabra "w" con una probabilidad P que es producto de dos probabilidades p_1 y p_2. Para cada tema, se calculan dos probabilidades p_1 y p_2.

- $p_1 = p(\frac{T}{D})$: la proporción de palabras en el documento y que actualmente están asignadas al tema T.
- $p_2 = p(\frac{w}{T})$: la proporción de asignaciones al tema t sobre todos los documentos que provienen de esta palabra w.

Después de varias iteraciones, se logra un estado estable donde el tema del documento y las distribuciones de los términos del tema son bastante buenos.

Se debe tener en cuenta que el algoritmo no tiene información externa sobre los temas encontrados y los artículos no están etiquetados con los temas o palabras claves (es un proceso de aprendizaje no supervisado). La distribución de temas encontrados surge mediante el cálculo de la estructura de temas ocultos que probablemente generan la colección de documentos observados.

1.6 Métricas de evaluación en modelos predictivos

El modelado predictivo funciona sobre el principio de retroalimentación constructiva: se construye un modelo, se obtiene retroalimentación de las métricas, se realiza mejoras y

se continua hasta lograr una precisión deseable. Las métricas de evaluación explican el rendimiento de un modelo; aspecto fundamental del aprendizaje automático por su capacidad para discriminar entre los resultados del modelo.

El objetivo no es construir un modelo predictivo y pensar que sea éste el mejor posible, es mejor crear y seleccionar un modelo que proporcione una alta precisión en los datos de muestra. La evaluación es importante para comprender la calidad del modelo o la técnica, para refinar los parámetros en el proceso iterativo de aprendizaje y para seleccionar el modelo o la técnica más aceptable de un conjunto dado de modelos o técnicas. Hay varios criterios para evaluar modelos para diferentes tareas y otros criterios que también pueden ser importantes, como la complejidad computacional o la comprensibilidad del modelo (Evaluation Measures for Data Mining Tasks, 2014). La elección de la métrica depende completamente del tipo de modelo y el plan de implementación del modelo.

Para problemas de clasificación es natural medir el rendimiento de un clasificador en términos de tasa de error. El clasificador predice la clase de cada instancia, si es correcto, se cuenta como "éxito", si no, es un error. La tasa de error es solo la proporción de los errores cometidos en todo el conjunto de instancias y mide el rendimiento general del clasificador. Se describen a continuación algunos de los métodos para la evaluación del rendimiento de un clasificador, los cuales se tomarán en cuenta en la presentación y discusión de los resultados de la propuesta de solución.

1.6.1 Matriz de confusión

Un modelo de clasificación binaria clasifica cada instancia en una de dos clases: una clase verdadera y una falsa. Esto da lugar a cuatro clasificaciones posibles para cada instancia: un verdadero positivo (TP, del inglés: *true positive*), un verdadero negativo (TN, del inglés: *true negative*), un falso positivo (FP, del inglés: *false positive*) o un falso negativo (FN, del inglés: *false negative*). Esta situación se puede representar como una matriz de confusión (también llamada tabla de contingencia) que se muestra en la figura (Fig. 10) (Evaluation Measures for Data Mining Tasks, 2014). La matriz de confusión es una presentación práctica de la precisión de un modelo con dos o más clases.

Confusion Matriz		Observed			
		True	False		
Model Predicted	True	True Positive (TP)	False Positive (FP)	Positve Predictive Value	TP/(TP+FP)
	False	Flase Negative (FN)	True Negative (TN)	Negative Predictive Value	TN/(FP+TN)
		Sensitivity	Specificity	**Accuracy** = (TP+TN)/(TP+FP+FN+TN)	
		TP/(TP+FN)	TN/(FP+TN)		

Fig. 10 Matriz de confusión y fórmulas de métricas de rendimiento

La matriz de confusión yuxtapone las clasificaciones observadas para un fenómeno (columnas) con las clasificaciones predichas de un modelo (filas). En general, en una matriz de confusión, las clases predichas se comparan con las clases reales. Cada fila de la matriz representa los resultados de la predicción para la clase correspondiente en esa fila, mientras que cada columna representa la clase real. En la figura (Fig. 10), las clasificaciones que se encuentran a lo largo de la diagonal principal de la tabla son las clasificaciones correctas, es decir, los verdaderos positivos y los verdaderos negativos; sin embargo, los otros campos significan errores de modelo. Para un modelo perfecto se analizan los campos verdadero positivo y verdadero negativo, los otros campos se establecerían en cero.

1.6.2 Exactitud

Una serie de métricas de rendimiento del modelo pueden derivarse de la matriz de confusión; la más común es la exactitud (del inglés, *accuracy*), aunque también existen otras métricas que son definidas a continuación y se visualizan sus fórmulas en la Fig. 10, como parte de una matriz de confusión:

- **Valor predictivo positivo (*precisión*)**: la proporción de casos positivos que se identificaron correctamente.
- **Valor predictivo negativo**: la proporción de casos negativos que se identificaron correctamente.
- **Sensibilidad (*recall*)**: la proporción de casos positivos reales que están correctamente identificados.

- **Especificidad**: la proporción de casos negativos reales que están correctamente identificados.

La exactitud (*accuracy*) de la clasificación es el número de predicciones correctas realizadas como una proporción de todas las predicciones realizadas. Esta es la evaluación más común de la métrica para problemas de clasificación, también es la más mal utilizada. Realmente solo es adecuado cuando hay un número igual de observaciones en cada clase (lo que rara vez es el caso) y que todas las predicciones y errores de predicción son igualmente importantes, lo que a menudo no es el caso (Brownlee, 2016).

1.6.3 Valor F1

El Valor-F (denominada también F-score o medida-F) en estadística es la medida de precisión que tiene un *test*. Se emplea en la determinación de un valor único ponderado de la precisión y la exhaustividad (*recall*) (Beitzel., 2006). Se suele emplear en la fase de pruebas de algoritmos de búsqueda y RI y clasificación de documentos (Li & Acero, 2008). La misma es calculable a partir de la siguiente fórmula (7):

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (7)$$

El puntaje de F1 es necesario cuando se desea buscar el equilibrio entre *precision* y *recall*, alcanzando su mejor valor en 1 (*precision* y *recall* perfectas) y el peor en 0.

Para generalizar esto a las clases múltiples, asumiendo que se tiene un clasificador *One-vs-All* (OvA) (Uno contra todo) o multi - clase, el puntaje F1 se puede calcular de dos formas diferentes: promedio micro (8) y promedio macro (9):

- En “micro” se calcularía el rendimiento, por ejemplo, la precisión, a partir de los positivos verdaderos individuales, negativos verdaderos, falsos positivos y falsos negativos del modelo de la clase k.

$$PRE_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k} \quad (8)$$

- En “macro”, se promedian los rendimientos de cada clase individual.

$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_k}{k} \quad (9)$$

Capítulo 2: Estado del Arte

Introducción

Según el Diccionario de la Real Academia Española:

- Texto es un enunciado o conjunto coherente de enunciados orales o escritos.
- Documento es un diploma, carta, relación u otro escrito que ilustra acerca de algún hecho; escrito en que constan datos fidedignos o susceptibles de ser empleados como tales para probar algo.
- Clasificar es ordenar o disponer por clases algo; dar carácter secreto o reservado a un documento.

La clasificación de textos y/o documentos es un campo en constante desarrollo que ha tomado gran importancia en distintas áreas del conocimiento humano. En la actualidad, existen grandes colecciones de documentos, artículos y recursos textuales que sirven distintos propósitos. Para mantener un control riguroso de tales colecciones es necesario emplear herramientas automáticas de análisis y procesamiento de textos. Por tal motivo, se realizan investigaciones orientadas a crear y mejorar las técnicas y metodologías usadas en la clasificación de textos y/o documentos.

A continuación, se realiza un análisis de algunos trabajos académicos, lo cual ha servido de fundamento y base para la presente investigación, ya que reflejan el uso de técnicas y herramientas para la clasificación de textos y documentos, aunque no se ajustan a las necesidades y características de la problemática que se aborda.

2.1 Análisis del trabajo: *Detección del engaño en notas de opinión a través de técnicas tradicionales de clasificación automática de textos*

El trabajo (Sánchez J., Villaseñor P., Escalante, & Montes y G., 2007) se centra en la detección de engaño en opiniones que fueron escritas con la intención de resumir o valorar una supuesta experiencia. El engaño está presente en múltiples actividades en las que el ser humano tiene la posibilidad, necesidad u obligación de expresarse verbalmente. Ejemplo de ello se puede apreciar en la web, donde los usuarios dejan plasmada una reseña, valoración u opinión sobre prácticamente todo. A menudo puede ser útil leer sobre la experiencia de otros para tomar una elección propia, y sabiendo esto a muchos les conviene falsear opiniones en busca de algún beneficio.

En el trabajo, se realiza un estudio del alcance de técnicas tradicionales usadas en clasificación automática de textos (bolsa de palabras, del inglés, *bag of words*) para la detección de engaño. Comúnmente las técnicas tradicionales funcionan adecuadamente en clasificación temática, pero los autores deciden conocer el rendimiento de dichas técnicas en una tarea intuitivamente no-temática. La colección empleada es un conjunto de notas en inglés de opiniones sobre hoteles, incluyendo notas verdaderas y falsas. Se realizaron experimentos utilizando la representación de bolsa de palabras con esquemas de pesado (binario, tf y tf-idf) y entrenando un clasificador probabilístico. Los resultados muestran que el engaño puede ser detectado con el enfoque tradicional.

Durante sus análisis se utiliza el modelo *Naïve Bayes Multinomial Updatable*, implementado en la plataforma *Weka*[3], el cual se ha usado satisfactoriamente en problemas de clasificación, fundamentalmente en clasificación temática de textos. Para el entrenamiento y validación del clasificador que se utiliza en este trabajo se emplea el corpus de opiniones sobre 20 hoteles de Chicago, en un total de 1600 opiniones, tomando en cuenta la información sobre la falsedad o veracidad de las opiniones, proponiéndose como trabajo futuro la inclusión de la polaridad. El pre - procesamiento de los textos se llevó a cabo con el uso de NLTK 3.0.

2.2 Análisis del trabajo: *KNN based Machine Learning Approach for Text and Document Mining*

La clasificación de textos utiliza varias herramientas de RI y Aprendizaje Automático. Un clasificador, capaz de clasificar correctamente los documentos que no se ven, para aprender, necesita ser entrenado con algunos documentos preclasificados de cada categoría, de modo que pueda generalizar el modelo que tiene, aprender de los documentos preclasificados y usar ese modelo para clasificar correctamente los documentos no vistos. En este trabajo (Bijalwan, Kumar, Kumari, & Pascual, 2014), primero categorizan los documentos usando el enfoque de Aprendizaje Automático basado en KNN (k-vecinos más cercanos) y luego devuelven los documentos más relevantes.

[3] Weka es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java. Es software libre distribuido bajo la licencia GNU-GPL.

A partir del experimento, la técnica de aprendizaje basada en KNN muestra la precisión máxima en comparación con la técnica de clasificación Naïve Bayes y *Term-Graph* para la extracción de texto o documentos. El inconveniente de KNN es que su complejidad de tiempo es alta, pero ofrece una mayor precisión que otras. Implementan *Term-Graph* con otros métodos en lugar del tradicional *Term-Graph* utilizado con AFOPT. Este híbrido muestra un mejor resultado que la combinación tradicional. Se hizo una aplicación de recuperación de información utilizando *Vector Space Model* para dar el resultado de la consulta ingresada por el cliente al mostrar el documento relevante.

2.3 Análisis del trabajo: *Tweets Classification Using Corpus Dependent Tags, Character and POS N-grams*

Los Perfiles de Autor centrados en los textos de Internet han estado creciendo durante los últimos años. Una de las razones es la gran cantidad de información que se produce cada minuto en las redes sociales o blogs. Estos textos de Internet tienen características propias que difícilmente pueden compararse con textos literarios, documentales o ensayos; esto se debe a la necesidad de tener una comunicación rápida y la libertad de publicar contenido no revisado.

A partir de dicha problemática, el trabajo (Gónzalez G., y otros, 2015) es parte de la tarea de creación de perfiles de autor en el concurso PAN[4] 2015. Los participantes tuvieron que predecir los rasgos de género, edad y personalidad de los usuarios de Twitter en cuatro idiomas diferentes (español, inglés, italiano y holandés). El enfoque tiene en cuenta las características estilísticas representadas por *N-gramas* de caracteres y *POS N-gramas* para clasificar los tweets. La idea principal del uso de *N-gramas* de caracteres es extraer la mayor cantidad de información posible codificada dentro del tweet (emoticones, inundaciones de caracteres, uso de letras mayúsculas, etc.). Los *POS N-gramas* se obtuvieron utilizando *Freeling* y ciertos tokens fueron re - etiquetados con etiquetas dependientes de Twitter. Los resultados obtenidos fueron muy satisfactorios; el puntaje de clasificación global fue de 83.46%.

Fue importante para los autores, señalar que, al comparar su enfoque con los otros dos mejores participantes, es un poco más lento debido al análisis gramatical realizado por *Freeling*. Además, que el uso de *N-gramas* de caracteres y *POS N-gramas*, es una

[4] El PAN es un laboratorio de evaluación sobre descubrimiento de plagio, autoría y uso indebido de software social, que se celebra en el marco de la conferencia CLEF

buena opción con textos densos debido a su capacidad de extracción. En el caso de los *N-gramas*, fue posible extraer emoticones, exagerar los signos de puntuación, usar mayúsculas y todo tipo de información emocional codificada en el tweet. Con *POS N-gramas*, en español e inglés se pudo capturar las series más representativas de dos y tres elementos gramaticales; en italiano y holandés, capturar los elementos gramaticales más frecuentes.

2.4 Análisis del trabajo: *Enhancing Semi-Supervised Text Classification using Document Summaries*

Tradicionalmente, la clasificación de textos se aborda mediante técnicas supervisadas. Además, una práctica común considerada en la línea de clasificación de textos es la aplicación de métodos de selección de funciones, que utilizan información estadística del conjunto de capacitación para identificar aquellos atributos que describen mejor los documentos entre las diferentes categorías. Sin embargo, bajo este paradigma, un problema importante es el alto costo involucrado en la recopilación de suficientes datos etiquetados para construir un modelo de clasificación eficaz, así como para realizar una selección de características adecuada.

Con el fin de superar dicha limitación, el trabajo (Villatoro T., Anguiano, Montes y G., Villaseñor P., & Ramírez de la Rosa, 2016) propone un enfoque alternativo para la clasificación de texto semi - supervisado, que se basa en una nueva estrategia para disminuir la sensibilidad al ruido contenido en los datos etiquetados por medio de la síntesis automática de texto. Su objetivo fue determinar la utilidad del resumen como una técnica de filtrado de ruido para mejorar la clasificación de texto semi - supervisada. Los resultados experimentales mostraron que el enfoque propuesto supera las técnicas tradicionales de clasificación de texto semi - supervisada. Además, los resultados también indican que el enfoque es adecuado para aprender de un solo ejemplo etiquetado por categoría.

En particular utilizaron el clasificador *Support Vector Machine* (SVM) dado que es especialmente adecuado para trabajar con conjuntos de datos con alta dimensionalidad. Para realizar los experimentos, emplearon la implementación de SVM incluida en el kit de herramientas de *Weka* con los parámetros predeterminados. Para validar las hipótesis, se realizó experimentos con el conjunto de datos R8; esta colección está formada por las ocho categorías más grandes del corpus Reuters-21578, cuyos documentos pertenecen a una sola clase.

2.5 Análisis del trabajo: *Minería de texto como una herramienta para la búsqueda de artículos científicos para la investigación*

El trabajo (Arias C., Mattos S., Heredia G., & Heredia V., 2016) se fundamenta en lo engorroso que se hace las búsquedas de datos que son de suma importancia para el desarrollo de un tema en especial o de interés, debido a las diversas bases de datos existentes y todo tipo de información textual, además de la gran cantidad de texto contenido en el mismo, el cual se tarda mucho tiempo en la búsqueda y la extracción de algunos conceptos. Por lo que se da importancia a la implementación de ciertas herramientas de minerías de texto que de manera eficaz y segura permiten llegar a ese objetivo.

Se exponen y explican las diversas maneras de cómo la minería de texto ayuda en la elaboración de un artículo de investigación o un estado del arte, teniendo en cuenta el tópico a explicar en dicho texto, se utilizan diferentes metodologías para implementar la minería de textos con el fin de analizar otros diferentes artículos potenciales, teniendo así, un mejor enfoque sobre el material cimiento de un nuevo proyecto o artículo. Se muestran pruebas realizadas en la herramienta de *software Weka* donde se evidencian los resultados que arroja el software de los artículos que se analizan por medio de este.

Además, se especifican algunos *softwares* que de manera automática implementan varias técnicas y metodologías aplicadas a la minería de textos para arrojar determinados resultados; seleccionadas debido a la utilización basada en su desarrollador, funcionamiento y popularidad.

Capítulo 3: Herramientas y tecnologías para el desarrollo de la propuesta de solución

Introducción

En el proceso de desarrollo para la implementación de los algoritmos es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten y mayores facilidades brinden para dar solución al problema de la investigación. Para la implementación de algoritmos para el análisis de artículos científicos, se adoptan las tecnologías y herramientas definidas por el autor de la presente investigación; debido a que este ambiente se ajusta a la necesidad de migrar progresivamente hacia plataformas y aplicaciones de código abierto, como parte de alcanzar la seguridad, invulnerabilidad e independencia tecnológica. En este capítulo se presentan algunas características del entorno que se ha decidido utilizar para el desarrollo de la tesis.

3.1 Lenguaje de programación para el procesamiento de textos

Un lenguaje de programación está diseñado para describir el conjunto de acciones consecutivas que un equipo de cómputo debe ejecutar. Es aquella estructura con cierta base sintáctica y semántica, impone distintas instrucciones a un programa de computadora. Es un modo práctico para dar instrucciones a un equipo de cómputo, que permite al desarrollador comunicarse con los dispositivos de *hardware* y *software* existentes (Lobos, 2005).

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Actualmente existen dos versiones de *Python* con soporte, la 2.7 y la rama 3.x de este lenguaje de programación. Estas dos versiones son incompatibles en muchos aspectos ya que cada una tiene funciones y métodos no compatibles con la otra. Se elige este lenguaje de programación por las siguientes características generales (Python, s.f.):

- Leer un buen programa en *Python* es como leer inglés. Permite concentrarse en la solución del problema en lugar de la sintaxis.
- Libre y Fuente Abierta: es un ejemplo de un FLOSS (*Free/Libre and Open Source Software*).
- Lenguaje de Alto Nivel: no hay que preocuparse por detalles de bajo nivel como, por ejemplo, el manejo de la memoria empleada para el programa.
- Portable: debido a su naturaleza de ser *Open Source*, *Python* ha sido portado a diversas plataformas. Se puede usar sobre *Linux, Windows, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE y PocketPC*.
- Orientado a Objetos: el programa está construido sobre procedimientos o funciones los cuales no son nada más que piezas de programa reutilizables.
- Librerías Extendidas: la librería estándar de *Python* es de hecho muy amplia. Puede ayudar a hacer varias cosas que involucran: expresiones regulares, generación de documentos, evaluación de unidades, pruebas, procesos, bases de datos, ftp, correo electrónico, XML, HTML, entre otras.

Además, *Python* permite realizar tareas de pre - procesamiento de textos con la utilización de algunas de sus herramientas las cuales se explican con posterioridad. Por tales motivos, para el desarrollo de la implementación del problema en cuestión se ha decidido utilizar la versión de *Python 3.7*.

3.2 Entorno de desarrollo integrado

Un entorno de desarrollo integrado o entorno de desarrollo interactivo (IDE, del inglés, *Integrated Development Environment*) es una herramienta de *software* dedicada exclusivamente al desarrollo de programas informáticos, ofrece una serie de complementos que facilitan el desarrollo ágil de *software*. Para el desarrollo de la propuesta en cuestión, se ha decidido utilizar el entorno de desarrollo *Jupyter Notebook* haciendo uso de la distribución de código abierto *Anaconda 3*, debido a las siguientes características (Jupyter, 2015):

- Amplía el enfoque basado en la consola para la computación interactiva, proporcionando una aplicación basada en web adecuada para capturar todo el

proceso de computación: desarrollar, documentar y ejecutar código, así como comunicar los resultados.

- Combina dos componentes: una aplicación web y documentos de cuaderno.
- La herramienta permite desarrollar código con varios lenguajes de programación, incluidos *Python*, *Julia*, *R*, *Haskell* y *Ruby*, de manera dinámica.
- Permite integrar en un mismo documento tanto bloques de código como texto, gráficas o imágenes.
- A menudo se utiliza para trabajar con datos, modelos estadísticos y aprendizaje automático.
- El uso de herramientas como *Anaconda* hace que sea fácil tener varias versiones de Python instaladas una al lado de la otra.

3.3 Herramientas para el procesamiento de textos

Para tareas de procesamiento de textos, se hace de vital importancia el uso de algunas de las herramientas comunes ya implementadas en *Python* para esta finalidad. Dichas herramientas poseen una colección de paquetes y objetos en *Python* muy adaptados para tareas del PLN. A continuación, se hace una breve explicación de algunas de estas herramientas las cuales sirven para la implementación del problema en cuestión.

3.3.1 NLTK

NLTK es una de las bibliotecas PLN más conocidas, potente y más utilizadas en el ecosistema de *Python*, útil para todo tipo de tareas, desde tokenización, hasta derivación, etiquetado de parte del habla y más. NLTK es una herramienta *Open Source* (NLTK, s.f.). Con el uso de esta herramienta también:

- Se puede examinar el contexto de un texto a partir de una simple lectura.
- Se puede ver cuáles son las palabras más usadas en el texto, o ver si cuantas veces aparece una palabra en un texto.
- Se puede obtener el contexto que comparten dos o más palabras.
- Ofrece soporte integrado para contar las diferentes palabras de un texto, así como realizar una distribución de frecuencias e ir anotando las veces que cada palabra aparece en el texto.

3.3.2 NumPy y Pandas

NumPy es una extensión de *Python* que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices (NumPy, 2019).

Pandas es una biblioteca de *software* escrita como extensión de *NumPy* para manipulación y análisis de datos para el lenguaje de programación *Python*. Es un *software* libre distribuido bajo la licencia BSD versión tres cláusulas (pandas, 2019). Características de la biblioteca:

- Tipo de datos *DataFrame* para manipulación de datos con indexación integrada.
- Herramientas para leer y escribir datos entre estructuras de dato en memoria y formatos de archivo variados.
- Alineación de dato y manejo integrado de datos faltantes.
- Reestructuración y segmentación de conjuntos de datos.
- Segmentación vertical basada en etiquetas, indexación elegante, y segmentación horizontal de grandes conjuntos de datos.
- Inserción y eliminación de columnas en estructuras de datos.
- Agrupación predefinida en la librería lo que permite realizar cadenas de operaciones dividir-aplicar-combinar sobre conjuntos de datos.
- Mezcla y unión de datos.

3.3.3 Spacy

spaCy es un marco en el entorno de PLN moderno y confiable que rápidamente se convirtió en el estándar para hacer PLN con *Python*, escrito en *Cython*. Sus principales ventajas son: velocidad, precisión y extensibilidad. Entre otras características, éste incluye (spacy, 2019):

- Índice de preservación de tokenización. En lugar de solo mantener las palabras, *spaCy* mantiene los espacios también. Esto es útil para situaciones en las que necesita reemplazar palabras en el texto original o agregar algunas anotaciones. Con la tokenización NLTK, no hay forma de saber exactamente dónde se

encuentra una palabra tokenizada en el texto original en bruto. *spaCy* conserva el vínculo entre la palabra y su lugar en el texto sin formato.

- Modelos para etiquetado POS, reconocimiento de entidad nombrada y análisis de dependencia.
- Posee visualizaciones fáciles y bonitas y soporta más de 33 idiomas.
- Extensible ya que puede trabajar con otras herramientas como: *Scikit-Learn, TensorFlow, gensim*.
- Viene con un modelo de vectores de palabras pretratados.

3.3.4 Scikit-learn

Scikit-learn es una biblioteca de aprendizaje automático, de *software* gratuito y código abierto, utilizable comercialmente - licencia BSD para el lenguaje de programación *Python*. Es una herramienta simple y eficiente para la minería de datos y el análisis de datos. Accesible para todos, y reutilizable en diversos contextos. Construido en *NumPy*, *SciPy* y *matplotlib*, así como para interacturas con estas bibliotecas numéricas y científicas (scikit-learn, scikit-learn: Machine Learning in Python, s.f.). *Scikit-learn* proporciona utilidades para extraer las características del texto: tokenización, conteo de ocurrencias de tokens y normalización, esquemas de pesado

El módulo *sklearn.feature_extraction* se puede usar para extraer características en un formato compatible con algoritmos de aprendizaje automático de conjuntos de datos que consisten en formatos como texto e imagen. Cuenta con varios algoritmos de clasificación, regresión y *clustering* (del español, agrupamiento), que incluyen *support vector machines*, *random forests*, *gradient boosting*, *k-means* y DBSCAN.

3.3.5 FreeLing

FreeLing es una suite de herramientas de análisis de lenguaje de código abierto, publicada bajo la Licencia Pública General *Affero* GNU de la *Free Software Foundation*. El proyecto *FreeLing* fue creado como un medio para poner a disposición de la comunidad los resultados de la investigación llevada a cabo en el grupo de investigación de PLN de la UPC (*Universitat Politècnica de Catalunya*) (FreeLing, s.f.).

FreeLing es una biblioteca de C++ que proporciona funcionalidades de análisis de lenguaje (análisis morfológico, detección de entidades nombradas, etiquetado de puntos

de interés, análisis, desambiguación del sentido de las palabras, etiquetado de roles semánticos, etcétera) para una variedad de idiomas (inglés, español, portugués, italiano, francés, alemán, ruso, noruego, catalán, gallego, croata, esloveno, asturiano, galés, entre otros). También proporciona un *front-end* de línea de comandos que se puede utilizar para analizar textos y obtener la salida en el formato deseado (XML, JSON, CoNLL). Está diseñado para ser utilizado como una biblioteca externa desde cualquier aplicación que requiera este tipo de servicios.

Entre sus principales servicios ofrecidos por la biblioteca *FreeLing*, se pueden citar:

- Tokenización de texto.
- División de oraciones.
- Análisis morfológico.
- Tratamiento de sufijos, retokenización de pronombres clíticos.
- Reconocimiento de palabras compuestas.
- Reconocimiento flexible de múltiples palabras.
- Separación por contracción.
- Predicción probabilística de categorías de palabras desconocidas.
- Codificación fonética.
- Búsqueda basada en SED para palabras similares en el diccionario
- Detección y clasificación de entidad nombrada.
- Reconocimiento de fechas, números, relaciones, moneda y magnitudes físicas (velocidad, peso, temperatura, densidad, etc.)
- Etiquetado de *POS.*
- Análisis de poca profundidad basado en gráficos.
- Anotación sensorial basada en *WordNet* y desambiguación.
- Análisis de dependencia basado en reglas.
- Análisis de dependencia estadística.
- Etiquetado semántico estadístico de roles.
- Resolución de la referencia.
- Extracción semántica de grafos.

Capítulo 4: Desarrollo de la propuesta de solución

Introducción

En este capítulo se muestra detalladamente cómo se procesa la información contenida en los documentos. Se explican los procesos a seguir para el análisis de artículos científicos generados por académicos de la Ciencia de la Computación de la UNAM; dándole solución al problema planteado, así como a los objetivos específicos del trabajo de tesis y las tareas de investigación descritos en la “Introducción” del presente trabajo.

4.1 Procedimiento general en el análisis de artículos

A partir del estudio realizado en el Capítulo 1 y Capítulo 2 abordados en el presente trabajo, se propone el siguiente procedimiento general en el análisis de artículos para el desarrollo de la propuesta de solución, mostrado en la figura (Fig. 11) siguiente:

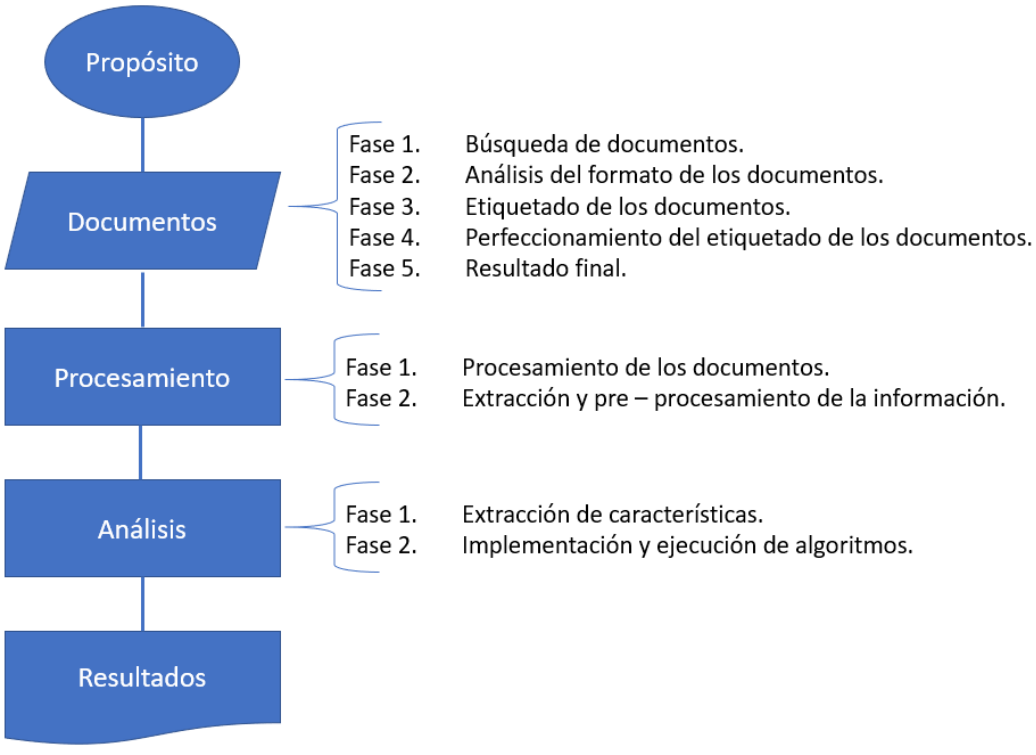

Fig. 11 Procedimiento general en el análisis de artículos

De esta propuesta se derivan las siguientes etapas:

Etapa 0. Propósito de la minería de documentos.

Etapa 1. Recuperación de los documentos.

Etapa 2. Procesamiento de los documentos.

Etapa 3. Análisis de la información.

Etapa 4. Presentación y discusión de resultados.

En las próximas secciones, se explica cómo ir cumpliendo cada una de las etapas mencionadas que resuelven el problema de la investigación planteado.

4.2 Etapa 0: Propósito de la minería de documentos

Esta etapa comienza con la detección de las necesidades para realizar el planteamiento de los objetivos y propósito del proyecto de minería en cuestión. Dicha etapa ha sido plasmada en la Introducción y el Capítulo 1, del presente trabajo. Etapa que fue necesaria para la identificación de las necesidades y planteamiento de los objetivos por el cual se está trabajando.

A partir de esto, se determinó plasmar las siguientes tareas de la investigación en el análisis de los documentos de académicos de la Ciencia de la Computación de la UNAM, de manera que se puedan ir respondiendo con el seguimiento de las etapas descritas anteriormente; las cuales se retoman a continuación:

T1. Clasificar a partir de su contenido un conjunto de documentos que se incorporen por área de conocimiento.

T2. Clasificar a partir de su resumen (del inglés, abstract) un conjunto de documentos que se incorporen por área de conocimiento.

T3. Clasificar a partir de sus palabras claves (del inglés, keywords) un conjunto de documentos que se incorporen por área de conocimiento.

T4. Identificar cuáles palabras claves son comunes a lo largo de diferentes áreas de conocimiento.

T5. Identificar cuáles temas son compartidos por diferentes áreas de conocimiento.

4.3 Etapa 1: Recuperación de los documentos

Esta etapa de recuperación de los documentos es una tarea clave y primordial, debido a que la representatividad y relevancia de los mismos con respecto al objetivo planteado impactará en la calidad de los resultados obtenidos. Como es de ocurrir en un proceso de minería, los resultados obtenidos son veraces en relación a los textos obtenidos, y su veracidad con respecto a la parcela de la realidad de nuestro interés dependerá de

lo representativos que sean los textos con respecto a dicha realidad. Para ello se han aplicado las siguientes fases:

Fase 1. Búsqueda de documentos.

En esta fase, se limita la búsqueda de los documentos para conformar el corpus, de acuerdo al tipo de temas e información con la que se está trabajando. Para dar solución al problema propuesto se tiene en cuenta que los artículos a buscar fueran escritos por académicos de la Ciencia de la Computación de la UNAM, específicamente de las seis áreas del conocimiento descritas en el Posgrado de Ciencias e Ingeniería de la Computación[5] (por sus siglas, PCIC). Se citan a continuación, las áreas del PCIC:

- Inteligencia Artificial.
- Redes y Seguridad en Cómputo.
- Computación Científica.
- Señales, Imágenes y Ambientes Virtuales.
- Ingeniería de Software y Bases de Datos.
- Teoría de la Computación.

Los documentos fueron extraídos a partir de buscadores como Google y *Google Scholar*, el repositorio de la Biblioteca del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas[6] de la UNAM (por sus siglas, IIMAS), el Repositorio Institucional RAD-UNAM[7] y los catálogos de bases de datos especializadas como: *IEEE Xplore*, *Web of Science* y *Scopus*.

Fase 2. Análisis del formato de los documentos.

Es de total importancia para la carga posterior de los documentos, que estos se encuentren en un formato apropiado, para que su manipulación se realice de una manera más simple, por lo que se tuvo en cuenta los siguientes aspectos:

- Los documentos no se encuentren cifrados, para poder extraer su información.
- Los documentos no sean de archivos escaneados ni de solo imágenes.

[5] Directorio de tutores. Listado por área de conocimiento. URL: http://www.pcic.unam.mx/tutoresarea.php
[6] Biblioteca del IIMAS: Acceso a recursos digitales. URL: https://www.iimas.unam.mx/biblioteca/acceso-recursos-digitales
[7] Repositorio Institucional RAD-UNAMURL: http://www.rad.unam.mx/

- Los documentos sean escritos en el idioma inglés.

Una vez extraído un conjunto de 766 documentos, se realiza un filtrado manual de los mismos, debido a que se identificaron algunos problemas durante la selección. En primera instancia, se identifica que algunos de los documentos estaban duplicados, por lo que se hace un filtrado a partir de sus títulos.

Por otro lado, se realiza un análisis comprensivo de la estructura interna de cada uno de los documentos, para lograr en la etapa posterior la extracción de la información necesaria, como: la extracción de los resúmenes (*abstract*) y sus palabras claves (*keywords*). Se tiene en cuenta la siguiente estructura: título, autores, resumen, palabras claves, introducción, desarrollo (varía el título de la sección o secciones en dependencia de la publicación del artículo), conclusiones y referencias bibliográficas.

Fase 3. Etiquetado de los documentos.

Como se explicaba en la Fase 1, se determinó la búsqueda de artículos escritos por académicos de la Ciencia de la Computación de la UNAM de las áreas del PCIC. Los documentos fueron etiquetados por estas áreas, enfocando la problemática en cuestión, en un problema de aprendizaje supervisado multi – clases, explicado en la sección “1.5 Técnicas en minería de textos” del Capítulo 1 de la presente investigación.

En conjunto con el análisis previo que se realiza en la fase anterior, también se tuvo en cuenta ciertos criterios para el etiquetado de los documentos por las áreas citadas. Se toma en cuenta: los autores que escribieron el artículo en conjunto al área al cuál pertenecen dentro del posgrado, el título del artículo, el contenido que abarca el resumen, las palabras claves y un vistazo en sentido general a todo el contenido del artículo en cuestión.

Posterior al trabajo realizado, quedaron un total de 280 documentos de 766 documentos anteriormente localizados.

Fase 4. Perfeccionamiento del etiquetado de los documentos.

Debido que el autor de la presente investigación, es quien realiza el etiquetado de los documentos, puede de alguna manera cometer errores, por cuestión humana y desconocimiento a la hora de etiquetar. Todo esto se explica debido a que, en ocasiones, por ejemplo: un artículo puede hablar de un cierto tema que, en efecto,

pertenece al área que se está otorgando; sin embargo, dentro del artículo, puede que se esté aplicando algún mecanismo o temas del cual pertenezca a otra de las áreas de clasificación que se tiene en cuenta. Para ello, se decide perfeccionar el etiquetado de los documentos, tomando en cuenta el criterio que a continuación se explica.

Se tuvo en cuenta tres anotadores o calificadores: el autor de la presente investigación, quien aporta las bases iniciales con el conjunto de documentos ya determinados en la fase anterior y, dos expertos más que conocen de los temas y contenidos de las áreas del conocimiento. A partir de sentar estas bases, se aplica el siguiente procedimiento:

- Se carga un archivo *.xlsx* llamado "observaciones". Las columnas de este archivo describen: los nombres de los documentos *.pdf* y otras 3 columnas más con la respectiva etiqueta dadas por los anotadores por cada documento, indicando los números del 0 al 6, donde el 6 indica que "No aplica a ningún área" y los restantes números, las áreas de conocimiento propuestas, a partir del mismo orden escrito anteriormente.
- Se calcula el valor de kappa de Cohen[8] (Cohen, 1960), identificado por la variable k, y se tiene en cuenta la interpretación del valor obtenido (Altman, 1991), la cual puede verse en la figura (Fig. 12).

Value of *K*	Strength of agreement
< 0.20	Poor
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Good
0.81 - 1.00	Very good

Fig. 12 Interpretación del valor de k.

Fuente: https://www.medcalc.org/manual/inter-rater_agreement.php

- El valor de k obtenido en primera instancia fue de 0.784, indicando un buen nivel de acuerdo entre los anotadores. Pero se procede a mejorar este valor:
  - Se elimina aquellos documentos que los tres anotadores no coincidieron en criterio.

[8] La kappa de Cohen: una estadística que mide el acuerdo entre anotadores. Esta función calcula la kappa de Cohen (Cohen, 1960), una puntuación que expresa el nivel de acuerdo entre anotadores sobre un problema de clasificación.

- Luego, se elimina aquellos documentos, con la condición que predominara una moda aritmética para el etiquetado con el número 6 (indicando que no aplica a ninguna de las áreas).
- Se vuelve a calcular el valor de k siendo de 0.807. El etiquetado final de los documentos se toma a partir de la moda aritmética (etiquetado mayoritario) dada por los anotadores por cada documento.

Se puede concluir que los documentos que quedan del conjunto total de seleccionados en la fase anterior, resultaron con un mejor etiquetado a partir del puntaje estadístico expresado por el nivel de acuerdo entre los anotadores sobre este problema de clasificación.

Fase 5. Resultado final.

Teniendo en cuenta la aplicación de las fases anteriores y la determinación del puntaje de kappa de Cohen obtenido, se concluye con un total de **272** documentos etiquetados de la siguiente manera (Fig. 13):

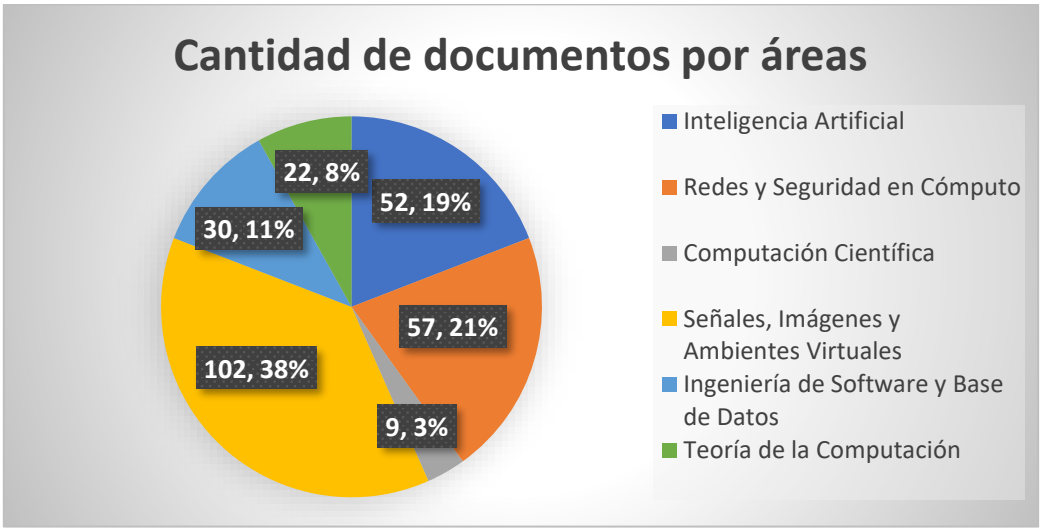

Fig. 13 Cantidad de documentos por áreas

Se puede apreciar un corpus desbalanceado, donde la mayor cantidad de artículos se encuentra en el área de “Señales, Imágenes y Ambientes Virtuales” con un total de 102 artículos y la de menor cantidad de artículos es del área de “Computación Científica” con un total de 9 artículos.

4.4 Etapa 2: Procesamiento de los documentos

Varios investigadores muestran que el pre - procesamiento es útil para varias tareas de PLN (Montes-y-Gómez, 2001) (Justicia de la T., 2017). En esta etapa se lleva a cabo el tratamiento del texto buscando que éste quede de una forma menos desestructurada. El pre - procesamiento es una de las etapas importantes para obtener una buena precisión de los métodos automatizados.

Para una posterior extracción de características y el análisis de la información, se aplican las siguientes fases para la extracción de la información y el pre – procesamiento de los textos, con el objetivo de mejorar la representación de n-gramas y reducir los errores de etiquetado POS:

Fase 1. Procesar los documentos

Se hace necesario antes de la extracción de la información en los documentos y el pre – procesamiento en los mismos, procesar los archivos PDF a TXT (texto plano). Es decir, transformar el conjunto de documentos de su formato original a un formato manipulable (en este caso formato ASCII, Código Estándar Estadounidense para el Intercambio de Información).

Se realiza un análisis de las diferentes bibliotecas que funcionan como OCR[9] en la forma de convertir archivos PDF a texto planos basados en el lenguaje de programación Python, como: *textract, PyPDF2* y *tika*. De estas bibliotecas, se decide utilizar *tika*[10], la cual analiza los archivos PDF de forma rápida, precisa y fácil de usar.

La secuencia de comandos recorre cada archivo PDF de las carpetas donde fueron almacenados los documentos y, para cada uno, carga su contenido, elimina las secuencias de varios espacios en blanco, tabuladores y saltos de línea los cuales fueron estandarizadas a un único espacio en blanco y, por último, se guardan a textos planos

[9] OCR (*Optical Character Recognition*) es un sistema computarizado de análisis que permite escanear un documento de texto en un fichero automatizado electrónicamente, que se puede editar con un procesador de textos en el ordenador.

[10] URL: https://github.com/chrismattmann/tika-python. Es un puerto de Apache Tika. Esta biblioteca necesita tener instalado en el sistema una versión de Java 7 o superior, debido a que *tika-python* inicia el servidor *Tika REST* en segundo plano

(*.txt*) en formato UTF-8[11] (sus siglas del inglés, *8-bit Unicode Transformation Format*). Se muestra en la figura (Fig. 14), como queda implementada la carga de los archivos.

Carga de los documentos

```
import glob
import os
from glob import glob
from tika import parser
import re

path_ia = glob("./RepositorioArticulosAcademicos/Artificial_intelligence/*.pdf")
path_nsc = glob("./RepositorioArticulosAcademicos/Networks_and_Security_in_Computation/*.pdf")
path_sc = glob("./RepositorioArticulosAcademicos/Scientific_Computing/*.pdf")
path_sive = glob("./RepositorioArticulosAcademicos/Signals_Images_and_Virtual_Environments/*.pdf")
path_sedb = glob("./RepositorioArticulosAcademicos/Software_engineering_and_databases/*.pdf")
path_tc = glob("./RepositorioArticulosAcademicos/Theory_of_computing/*.pdf")
articulos = [path_ia, path_nsc, path_sc, path_sive, path_sedb, path_tc]

def save_pdf_2_txt():
    cont = 0
    for i in range(len(articulos)):
        for j in range(len(articulos[i])):
            path = articulos[i][j]
            texto = read_pdf_file(path)
            spath = path.split("\\")
            nfile = spath[1].split(".")
            if j == 0:
                if os.path.isdir(spath[0] + '/text') == False:
                    os.makedirs(spath[0] + '/text')
            with open(spath[0] + '/text/' + nfile[0] + '.txt', "wb") as f:
                f.write(texto.encode("utf-8"))
                f.close()
            cont += 1
    print("He procesado todos los PDF a TXT. Total: ", cont)
    return

def read_pdf_file(filename):
    raw = parser.from_file(filename)
    text = " ".join(raw['content'].strip().split('\n'))
    text = re.sub(r" +", " ", re.sub(r"\t", " ", re.sub(r"\n+", " ", text)))
    return str(text)

save_pdf_2_txt()
```

```
He procesado todos los PDF a TXT. Total:  272
```

Fig. 14 Implementación de la carga de los archivos PDF a TXT

Para las tareas de clasificación de documentos con base al contenido, resumen y palabras claves respectivamente, en función de las **tareas de la investigación T1, T2 y T3**, se realiza una distribución automatizada del conjunto de documentos que se tiene. La distribución es para separar el conjunto de documentos en dos subconjuntos: un conjunto de entrenamiento y un conjunto de prueba. Esto nos sirve, para posteriormente medir el rendimiento del clasificador, debido a que es un problema de clasificación de aprendizaje supervisado. Se utiliza una de las funciones de la biblioteca *Scikit-learn*, que permite del corpus completo, dividirlo en un 75% para el conjunto de entrenamiento y en un 25% para el conjunto de prueba. En la tabla (Tabla 1) se muestra cómo queda la distribución del corpus.

[11] UTF-8 (*8-bit Unicode Transformation Format*) es un formato de codificación de caracteres Unicode e ISO 10646 utilizando símbolos de longitud variable (de 1 a 4 bytes por carácter Unicode) y capaz de representar cualquier carácter Unicode.

Tabla 1 Distribución del corpus: conjunto de entramiento y conjunto de prueba

Etiqueta	Áreas	Conjunto de entrenamiento	Conjunto de prueba	Totales
0	Inteligencia Artificial	41	11	52
1	Redes y Seguridad en Cómputo	43	14	57
2	Computación Científica	7	2	9
3	Señales, Imágenes y Ambientes Virtuales	73	29	102
4	Ingeniería de Software y Bases de Datos	22	8	30
5	Teoría de la Computación	18	4	22
	Total de documentos	**204**	**68**	**272**

Fase 2. Extracción y pre – procesamiento de la información

Debido a que los artículos académicos que se encuentran publicados se escriben con formalidad y lleva todo un proceso de revisión por parte del comité de expertos de las revistas y/o los congresos donde se publican, el pre – procesamiento para los mismos es diferente con respecto a un corpus compuesto por datos de redes sociales, por ejemplo.

En función de la **tarea de la investigación T1** para la clasificación de los documentos a partir de su contenido, se aplican los siguientes pasos:

1. Todas las palabras se transforman en minúsculas: evita tener múltiples copias de las mismas palabras.
2. Eliminación de palabras funcionales (o palabras comunes o vacías).

Las palabras vacías es un listado de términos (preposiciones, determinantes, pronombres, etcétera) considerados de escaso valor semántico, que cuando se identifican en un documento se eliminan, sin considerarse términos índices para la colección de textos a analizar. No aportan ningún significado al texto. La supresión de todos estos términos evita los problemas de ruido documental y supone un considerable ahorro de recursos, ya que, aunque se trata de un número relativamente reducido de elementos tienen una elevada tasa de frecuencia en los documentos.

3. Eliminación de signos de puntuación, ya que no agrega ninguna información adicional al tratar los datos de texto; eliminar todos los casos ayuda a reducir el tamaño de los datos de entrenamiento y prueba.
4. Las secuencias de varios espacios en blanco, tabuladores y saltos de línea se estandarizan a un único espacio en blanco.

La siguiente tabla (Tabla 2) presentan las estadísticas referentes a la longitud promedio de los documentos (número de caracteres promedio y número de palabras promedio) por cada una de las áreas etiquetadas, antes y después del pre – procesamiento; se puede reflejar que al realizar el pre – procesamiento hay una disminución elevada de términos que no son necesarios posteriormente procesar.

Tabla 2 Longitud promedio de los documentos antes y después del pre - procesamiento

		Longitud promedio de los documentos			
Etiqueta	**Áreas**	**Antes del pre - procesamiento**		**Después del pre - procesamiento**	
		Caracteres	**Palabras**	**Caracteres**	**Palabras**
0	Inteligencia Artificial	43518.53	6441.32	35269.0	4637.71
1	Redes y Seguridad en Cómputo	28310.31	4436.84	24541.92	3306.12
2	Computación Científica	34712.0	5856.66	30887.0	4304.11
3	Señales, Imágenes y Ambientes Virtuales	33814.87	5089.17	29385.02	3877.67
4	Ingeniería de Software y Bases de Datos	35190.46	5259.86	29099.56	3754.33
5	Teoría de la Computación	38673.27	6494.09	37733.77	4862.27

Para la extracción de los resúmenes de cada uno de los artículos y luego su posterior pre – procesamiento, en función de la **tarea de la investigación T2** para la clasificación de los documentos a partir de su resumen, se tuvo en cuenta las anotaciones realizadas durante la “Etapa 1: Recuperación de los documentos”, en su “Fase 2. Análisis del formato de los documentos”. Se citan algunos de los patrones detectados en dicha fase:

ABSTRACT This paper introduces… are depicted in this paper. Keywords genetic…

Abstract: In this article, … gray level and low contrast images. Key words: color metrics…

Abstract - An algorithm is presented… Index terms – Boundary tracking, …

Abstract – In this paper we present … are addressed as well. Keywords – Web based …

SUMMARY We present a scalable … the state-of-the-art methods. key words: object …

Abstract. Starting with only … algebras described above. Key words. Hopf algebras, …

Abstract. Content-based image … acquisition equipment. KeyWords: Content-based …

(se entiende como "…" a un conjunto de letras y símbolos que componen el texto)

Se aplican los siguientes pasos para la extracción y pre – procesamiento de los resúmenes:

1. Carga de los documentos, posteriormente guardados a archivos planos (*.txt*).
2. Se transforma todo el texto a minúsculas.
3. A partir de los patrones detectados, se implementan expresiones regulares para la extracción del resumen; las cuales se exponen a continuación:

```
pattern1 = r'(?:abstract|summary)'
pattern2 = r'(?:résumé|resumen|key word[s]?|keyword[s]?|index
                term[s]?|key_word[s]?|key-word[s]?)'
pattern3 = r'(?:[\w\W\s\S]*)'
# pattern1 + pattern3 + pattern2
```

4. Luego, se cambia por espacio en blanco lo encontrado con la unión de $pattern2 + pattern3$, para que solo quede el contenido del resumen.
5. Se guarda a texto plano (*.txt*) el contenido encontrado para su posterior manejo.
6. Para su pre – procesamiento: se eliminan las palabras funcionales, los signos de puntuación, y estandarizar a un único espacio en blanco, las secuencias de varios espacios en blanco, tabuladores y saltos de línea.

La siguiente tabla (Tabla 3) presenta las estadísticas referentes a la longitud promedio (número de caracteres promedio y número de palabras promedio) de los resúmenes, luego de su extracción, por cada una de las áreas etiquetadas, antes y después del pre - procesamiento:

Tabla 3 Longitud promedio de los resúmenes antes y después del pre - procesamiento

Etiqueta	Áreas	Longitud promedio de los resúmenes			
		Antes del pre - procesamiento		Después del pre - procesamiento	
		Caracteres	Palabras	Caracteres	Palabras
0	Inteligencia Artificial	1036.11	154.28	788.05	94.23
1	Redes y Seguridad en Cómputo	916.85	137.15	712.92	86.43
2	Computación Científica	976.11	145.33	769.55	96.88
3	Señales, Imágenes y Ambientes Virtuales	1056.34	156.52	821.99	99.78
4	Ingeniería de Software y Bases de Datos	716.6	105.03	566.03	67.3
5	Teoría de la Computación	852.81	128.90	656.77	82.5

De manera similar a la extracción y pre – procesamiento del resumen por documento, se procede para la extracción de las palabras claves, en función de la **tarea de la investigación T3** para la clasificación de los documentos a partir de sus palabras claves. Para este caso se detectaron algunos de los siguientes patrones:

Keywords: Logic program transformation, … , inductive logic programming. * Address for correspondence: Instituto de Investigaciones … 1. Introduction The unfolding…

Keywords: parallel class; … ; pruning 2000 AMS Subject Classification: 05B05; … 1 Introduction A design is …

Keywords: Software engineering, , … ,KUALI-BEH Mathematics Subject … 68-N30 1 INTRODUCTION Precisely specifying…

Keywords: chain codes; … ; voxel-based surfaces. Paper 150594 received May 8, … 1 Introduction Several methods …

Keywords: computerized tomography, … , superiorization (Some figures … online journal) 1. Introduction In a tupical x-ray …

Key words: marching, … , retinal images * Corresponding author … 1 Introduction Image …

(se entiende como "…" a un conjunto de letras y símbolos que componen el texto)

Keywords: Reasoning with Preference, … , Robot Golem III 1. Introduction Service robots…

Keywords 4E Cognition, …, historical background Handling Editor: … I. Introduction According …

Index terms ---- Humanoid, soccer, … , position. I. INTRODUCION The RobCup …

Keywords – FPGAs; State Machines; … ; Genetic Algorithms. I. INTRODUCTION In mobile …

key words: object discovery, … , agglomerative clustering 1. Introduction In most object …

Keywords: Ultra-Available System, … , Triple Modular Redundancy. Resumen Este estudio … 1 Introduction Safety-critical …

Keywords: Gesture recognition, … , human-machine interaction. 1 Introduction Visual…

Se aplican los siguientes pasos para la extracción y pre – procesamiento de las palabras claves:

1. Se carga los documentos, posteriormente guardados a archivos planos (*.txt*).
2. Todo el texto se transforma a minúsculas.
3. A partir de los patrones detectados se implementan expresiones regulares para la extracción de las palabras claves; los cuales se exponen a continuación:

```
pattern3 = r'(?:[\w\W\s\S]*)'
pattern4 = r'(?:key word[s]?|keyword[s]?|index term[s]?|key_word[s]?|key-word[s]?)'
pattern5 = r'(?:[\W]?[\s]*resumen|[\W]?[\s]*[1|i|a]?[.|_|-
        |\s]+introduct[ion]?|[\W]?[\s]*(?:(?:20)\d\d)[\s]*[ams|mathematics subject])'
# pattern4 + pattern3 + pattern5
# las restantes expresiones regulares
pattern6 = r'(?:[\W]*)'
pattern7 = r'(?:ams|mathematics subject|paper|fecha|received|handling editor|some
        figures|article type|running head|pacs|address all
correspondence|corresponding
        author|contacts|chinese library classification|address for
        correspondence|correspondence|mailto)'
pattern8 = r'(?:[,|;])'
pattern9 = r'(?:^(b\'|b"))'
```

4. Luego, se cambia por espacio en blanco lo encontrado con la unión de: $pattern5 + pattern3$, $pattern4 + pattern6$, $pattern7 + pattern3$ y $pattern9$, para que solo queden las palabras claves.
5. Se guarda a texto plano (*.txt*) el contenido encontrado para su posterior manejo.

6. Para su pre – procesamiento: se eliminan las palabras funcionales, los signos de puntuación, y estandarizar a un único espacio en blanco, las secuencias de varios espacios en blanco, tabuladores y saltos de línea.

La siguiente tabla (Tabla 4) presenta las estadísticas referentes a la longitud promedio (número de caracteres promedio y número de palabras promedio) de las palabras claves, luego de su extracción, por cada una de las áreas clasificadas, antes y después del pre - procesamiento:

Tabla 4 Longitud promedio de las palabras claves antes y después del pre - procesamiento

		Longitud promedio de las palabras claves			
Etiqueta	**Áreas**	**Antes del pre - procesamiento**		**Después del pre - procesamiento**	
		Caracteres	**Palabras**	**Caracteres**	**Palabras**
0	Inteligencia Artificial	84.98	9.32	83.19	9.55
1	Redes y Seguridad en Cómputo	76.22	8.43	76.43	8.82
2	Computación Científica	84.22	9.44	83.22	10.0
3	Señales, Imágenes y Ambientes Virtuales	92.40	10.36	91.29	10.62
4	Ingeniería de Software y Bases de Datos	74.63	8.3	72.56	8.66
5	Teoría de la Computación	92.04	9.81	89.63	9.95

En función de la **tarea de la investigación T4** para identificar las palabras claves más comunes a lo largo de las áreas de conocimiento, se decide unir en un único archivo *.txt* todo el conjunto de palabras claves por áreas, por lo cual se aplican los siguientes pasos:

1. Una vez que se tiene el conjunto de palabras claves por documentos de una misma área, se sustituye por salto de línea lo encontrado en el patrón 8 ($pattern8$), esto ayuda a separar cada palabra clave dentro del conjunto de palabras claves escritas por el académico.
2. Agrupar por cada documento de una misma área cada palabra clave aplicando el paso anterior.
3. Guardar en un texto plano *.txt* el conjunto de palabras claves por cada área separadas por salto de línea.
4. Para su pre – procesamiento se realiza el proceso de tokenización al conjunto de palabras claves por las diferentes áreas de conocimiento, explicado en la

sección “1.4 Tareas para el procesamiento de texto” del Capítulo 1 de la presente investigación. Se utiliza la función de *wordpuntc_tokenize()* de la biblioteca NLTK, esta función almacena los signos de puntuación como tokens diferentes.

5. Posteriormente se eliminan las palabras funcionales (extraídas del módulo *stopwords* de NLTK para el idioma inglés) y los signos de puntuación.

En función de la **tarea de la investigación T5** para los temas compartidos por diferentes áreas, se utiliza la función de *CountVectorizer* de la biblioteca *Scikit-learn*, siendo un proceso de vectorización que sirve para convertir una colección de documentos de texto en vectores de características numéricas. Se emplea *CountVectorizer* para:

1. Encontrar tokens de más de tres letras.
2. Eliminar palabras funcionales.
3. Convertir los tokens a minúsculas.
4. Eliminar tokens que aparecen en al menos del 25% del total de documentos.
5. Eliminar tokens que no aparecen en más del 50% de los documentos.

4.5 Etapa 3: Análisis de la información

En esta etapa, se aplica lo analizado en las secciones “1.4 Tareas para el procesamiento de texto” y “1.5 Técnicas en minería de textos” del Capítulo 1 de la presente investigación.

Fase 1. Extracción de características

Se expone a continuación el uso de algunos criterios utilizados como características. Estas características ayudan a las técnicas de clasificación, en función de las **tareas de la investigación T1, T2 y T3**, obtener un mayor porcentaje de exactitud a la hora de evaluar sus predicciones en función de realizar la clasificación. La detección de N-gramas consiste en la identificación de aquellas palabras que suelen aparecer juntas (palabras compuestas, nombres propios, etcétera) con el fin de tratarlas como una sola unidad conceptual.

- **N-gramas de caracteres:** Son capaces de detectar la composición morfológica de una palabra (Kulmizev, y otros, 2017). Para tareas de PLN, donde es probable que muchas palabras estén mal escritas, los n-gramas de caracteres son especialmente poderosos (Sanchez-Perez, Markov, Gómez-Adorno, & Sidorov,

2017) para detectar patrones en dichas faltas de ortografía (Kulmizev, y otros, 2017). Para este enfoque las variaciones de n van de 2 a 5.

- **N-gramas de palabras:** Capturan la identidad de una palabra y sus posibles vecinos (Kulmizev, y otros, 2017). Se encuentra en los experimentos la combinación de la palabra n-gramas con n variando de 1 a 5.
- **N-gramas de etiquetas POS:** Para la obtención del etiquetado POS de los documentos, se hace uso del servicio ofrecido por la biblioteca *FreeLing*, explicado en la sección "3.3 Herramientas para el procesamiento de textos" del Capítulo 3. Se experimentan con varias combinaciones de POS n-gramas en los datos de 1 a 5.
- **N-gramas con saltos de palabras (*skip-grams*):** Son un enfoque relativamente nuevo en el PLN, más notable por su efectividad para aproximar el significado de las palabras en los modelos de espacio vectorial (Mikolov, Chen, Corrado, & Dean, 2013). Se realizaron grupos de 2 palabras con saltos de 1 a 4 palabras.
- **N-gramas de palabras funcionales (*stop-words*):** A pesar del escaso valor semántico que contienen las palabras funcionales, en muchos casos tienen una elevada tasa de frecuencia en los documentos; en ocasiones hay escritos que contienen un exceso de ellas, lo cual podría ser un aspecto característico a la hora de clasificar textos. Previo al pre - procesamiento del corpus realizado en la Etapa 2, se realiza n-gramas de *stopwords*, tomando para el lenguaje inglés del NLTK, la variación de tokens de 1 a 4.
- **N-gramas de símbolos de puntuación:** Con esta característica se pretende determinar a partir del sistema de símbolos, lo cual contribuye a dar coherencia y cohesión al texto escrito, detectar ciertos patrones en el análisis de los escritos académicos por las diferentes áreas de conocimiento. Previo al pre - procesamiento del corpus realizado en la Etapa 2 se realiza n-gramas con variación de símbolos de 1 a 4.

Fase 2. Implementación y ejecución de algoritmos

En función de las **tareas de la investigación T1, T2 y T3**, se utilizan los algoritmos implementados en la biblioteca *Scikit-learn* como: *Multinomial Naïve Bayes*, *Logistic Regression* y *Linear Support Vector Classification* (LinearSVC), explicados en la sección "1.5 Técnicas en minería de textos" del Capítulo 1 de la presente investigación.

La configuración final del sistema implementa un esquema multi - clase, considera solo aquellas características que ocurren en al menos 50 documentos en el corpus, además de la incorporación de las características antes expuestas para alimentar a los algoritmos de clasificación escogidos. Se muestra en la figura (Fig. 15) un fragmento de código del método “clasificador”, donde se refleja lo antes expuesto.

Clasificador

```
#Leyenda: cn:character n-grams, wn:word n-grams, pn:pos n-grams, sn:skipgrams n-grams,
# fn:stop words n-grams, sp:punctuacion simbol n-gramas
def clasificador(X_train, X_test, y_train, y_test, cn, wn, pn, sn, fn, sp):
    start_time = time.time()
    print('Reading file')
    print(' - Extracting features')
    train_features, dicOfFeatures = process_texts(X_train, cn, wn, pn, sn, fn, sp)
    vectorizer = CountVectorizer(lowercase=True, min_df=50, tokenizer=lambda x: x.split('_&%$_'))
    X_train_vectorized = vectorizer.fit_transform(train_features)

    ###### Clasificador
    print(' - Training Classifier')

    ######Entrenar clasificador#########
    clf1 = MultinomialNB()
    clf2 = LogisticRegression()
    clf3 = LinearSVC(C=10000, random_state=0)

    clf1.fit(X_train_vectorized, y_train)
    clf2.fit(X_train_vectorized, y_train)
    clf3.fit(X_train_vectorized, y_train)

    ###### Test ########################
    print ('Reading Test files')
    print(' - Extracting Test features')
    test_features, dicOfFeaturesTest = process_texts(X_test, cn, wn, pn, sn, fn, sp)
    X_test_vectorized = vectorizer.transform(test_features)

    # Predicting Test
    print(' - Predicting Test')
    predictions1 = clf1.predict(X_test_vectorized)
    predictions2 = clf2.predict(X_test_vectorized)
    predictions3 = clf3.predict(X_test_vectorized)

    ##### Evaluation metrics #############
    print('\t', 'Multinomial NB - F1_score', f1_score(y_test, predictions1, average='macro'))
    print('\t', 'Logistic Regression - F1_score', f1_score(y_test, predictions2, average='macro'))
    print('\t', 'Lineal SVC - F1_score', f1_score(y_test, predictions3, average='macro'))

    return
```

Fig. 15 Fragmento de código del método "clasificador"

En función de la **tarea de la investigación T4**, se utilizan los algoritmos de *stemming*: el algoritmo de Porter y Porter 2, explicados en la sección “1.4 Tareas para el procesamiento de textos” del Capítulo 1 de la presente investigación. Por último, para ambos casos, se calcula las palabras más frecuentes y su frecuencia, y el tiempo de ejecución del uso de ambos algoritmos. Se muestra en la figura (Fig. 16) un fragmento del código de implementación de estos algoritmos para la identificación de las palabras claves más comunes a lo largo de las diferentes áreas de conocimiento.

```python
import nltk
import time
from nltk import *
from nltk import wordpunct_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem.snowball import SnowballStemmer

def common_words_Porter(path):
    start_time = time.time()
    with open(path, 'r', encoding='utf8') as file:
        pal = file.read().encode('utf-8')
        file.close()
    tokens = [word for word in wordpunct_tokenize(str(pal))]
    stopw = [w.encode('utf-8') for w in stopwords.words('english')]
    punc_lit = [u'.', u'[', ']', u',', u';', u'', u')', u'),', u' ', u'(']
    stopw.extend(punc_lit)
    words = [token for token in tokens if token not in stopw]
    porter_stemmer = PorterStemmer()
    stemmers = [porter_stemmer.stem(word) for word in words]
    final = [stem for stem in stemmers if stem.isalpha() and len(stem) > 1]
    fdist = nltk.FreqDist(final)
    print('elapsed time:', time.time() - start_time)
    return print(fdist.most_common(10))

def common_words_Porter2(path):
    start_time = time.time()
    with open(path, 'r', encoding='utf8') as file:
        pal = file.read().encode('utf-8')
        file.close()
    tokens = [word for word in wordpunct_tokenize(str(pal))]
    stopw = [w.encode('utf-8') for w in stopwords.words('english')]
    punc_lit = [u'.', u'[', ']', u',', u';', u'', u')', u'),', u' ', u'(']
    stopw.extend(punc_lit)
    words = [token for token in tokens if token not in stopw]
    sSnowball_stemmer = SnowballStemmer('english')
    stemmers2 = [sSnowball_stemmer.stem(word) for word in words]
    final2 = [stem for stem in stemmers2 if stem.isalpha() and len(stem) > 1]
    fdist2 = nltk.FreqDist(final2)
    print('elapsed time:', time.time() - start_time)
    return print(fdist2.most_common(10))
```

Fig. 16 Fragmento de código de los algoritmos para la identificación de las palabras claves más comunes por área

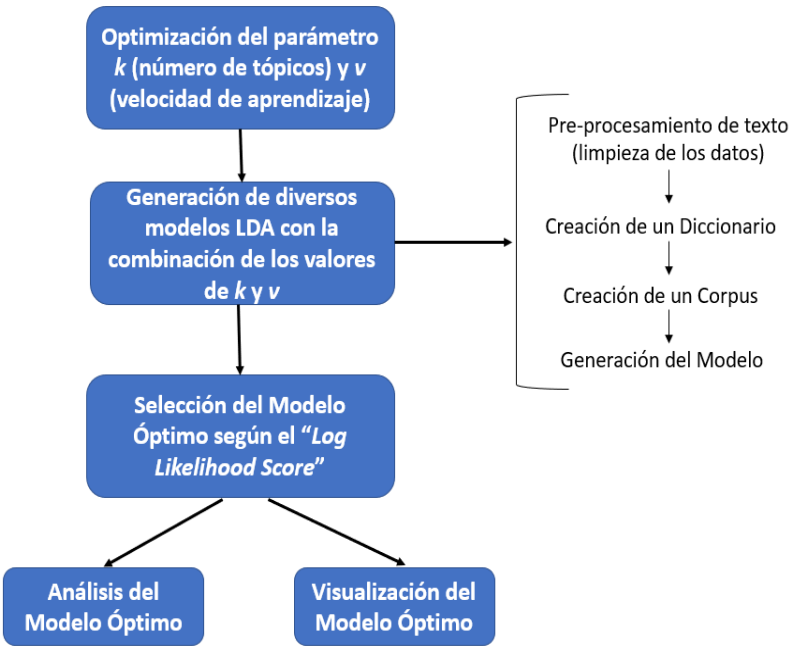

Fig. 17 Pasos a seguir para el cumplimiento del objetivo específico D5

En función de la **tarea de la investigación T5**, para solucionar el objetivo en cuestión se utiliza técnicas de aprendizaje no supervisado, explicado en la sección "1.5 Técnicas

en minería de textos" del Capítulo 1 de la presente investigación. La técnica que responde a la solución del objetivo es la de *Latent Dirichlet Allocation* (LDA) para el modelado de tópicos. Se puede observar en la figura (Fig. 17) los pasos que se sigue para llegar a la solución del mismo.

Como se observa en la figura (Fig. 17), se decide elegir el modelado óptimo, a partir de construir diversos modelos LDA con todas las posibles combinaciones de parámetros. Los parámetros que se tienen en cuenta son: un conjunto de número de tópicos (2, 3, 4, 5, 7, 10, 13, 15, 20) y el radio de velocidad de aprendizaje[12] (0.5, 0.7, 0.9). Para la selección del modelo óptimo se tiene en cuenta los valores de *Log-likelihood*[13], debido a que desea maximizar la probabilidad de registro, el valor más alto es mejor.

Se muestran en las figuras (Fig. 18 y Fig. 19) el fragmento del código de implementación para el modelado de tópicos a lo largo de las diferentes áreas de conocimiento. Se refleja los pasos que se tuvieron en cuenta a seguir para la selección del mejor modelo, así como la visualización de sus resultados.

[12] El parámetro *learning_decay* (velocidad de aprendizaje) es un parámetro dentro del método de modelado de tópico LDA que controla la tasa de aprendizaje en el método de aprendizaje en línea. El valor debe establecerse entre (0.5, 1.0] para garantizar la convergencia asintótica.

[13] *Log-likelihood*: en estadística, la función de verosimilitud (o, simplemente, verosimilitud o probabilidad logarítmica) es una función de los parámetros de un modelo estadístico que permite realizar inferencias acerca de su valor a partir de un conjunto de observaciones. Los valores de probabilidad logarítmica no se pueden usar solos como un índice de ajuste porque son una función del tamaño de la muestra, pero se pueden usar para comparar el ajuste de diferentes coeficientes. Debido a que desea maximizar la probabilidad de registro, el valor más alto es mejor.

```python
def topic_model(path, area):
    doc_txt = read_txt_files(glob(path + 'text/*.txt'))
    min_doc = round(len(doc_txt)*0.25)
    vectorizer = CountVectorizer(strip_accents = 'unicode', stop_words = 'english', lowercase = True,
                                 token_pattern = r'\b[a-zA-Z]{3,}\b', max_df = 0.5, min_df = min_doc)
    data_vectorized = vectorizer.fit_transform(doc_txt)
    #Materialize the sparse data
    data_dense = data_vectorized.todense()
    n_topics = [2, 3, 4, 5, 7, 10, 13, 15, 20]
    # Define Search Param
    search_params = {'n_topics': n_topics, 'learning_decay': [.5, .7, .9]}
    # Init the Model
    lda = LatentDirichletAllocation()
    # Init Grid Search Class
    model = GridSearchCV(lda, param_grid=search_params)
    # Do the Grid Search
    model.fit(data_vectorized)
    # Best Model
    best_lda_model = model.best_estimator_
    # Model Parameters
    print("Best Model's Params: ", model.best_params_)
    # Log Likelihood Score
    print("Best Log Likelihood Score: ", model.best_score_)
    # Perplexity
    print("Model Perplexity: ", best_lda_model.perplexity(data_vectorized))
    # Get Log Likelyhoods from Grid Search Output
    log_likelyhoods_5 = [round(gscore.mean_validation_score) for gscore in model.grid_scores_
                         if gscore.parameters['learning_decay']==0.5]
    log_likelyhoods_7 = [round(gscore.mean_validation_score) for gscore in model.grid_scores_
                         if gscore.parameters['learning_decay']==0.7]
    log_likelyhoods_9 = [round(gscore.mean_validation_score) for gscore in model.grid_scores_
                         if gscore.parameters['learning_decay']==0.9]
    # Show graph
    plt.figure(figsize=(12, 8))
    plt.plot(n_topics, log_likelyhoods_5, label='0.5')
    plt.plot(n_topics, log_likelyhoods_7, label='0.7')
    plt.plot(n_topics, log_likelyhoods_9, label='0.9')
    plt.title("Choosing Optimal LDA Model")
    plt.xlabel("Num Topics")
    plt.ylabel("Log Likelyhood Scores")
    plt.legend(title='Learning decay', loc='best')
    plt.show()
```

Fig. 18 Fragmento de código de la implementación del modelado de tópicos (Parte 1)

```python

    n_words=15
    tf_feature_names = vectorizer.get_feature_names()
    print_top_words(best_lda_model, tf_feature_names, n_words)
    topic_keywords = show_topics(vectorizer, best_lda_model, n_words)
    # Topic - Keywords Dataframe
    df_topic_keywords = pd.DataFrame(topic_keywords)
    df_topic_keywords.columns = ['Word '+str(i) for i in range(df_topic_keywords.shape[1])]
    df_topic_keywords.index = ['Topic '+str(i) for i in range(df_topic_keywords.shape[0])]
    df_topic_keywords

    # to visualize the LDA model with pyLDAvis
    pyLDAvis.enable_notebook()
    panel = pyLDAvis.sklearn.prepare(best_lda_model, data_vectorized, vectorizer, mds='tsne')
    pyLDAvis.save_html(panel, path +'lda_'+ area + '.html')
    return panel

# Show top n keywords for each topic
def show_topics(vectorizer, lda_model, n_words):
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords

def print_top_words(model, feature_names, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += " ".join([feature_names[i]
                             for i in topic.argsort()[:-n_top_words - 1:-1]])
        print(message)
    print()
```

Fig. 19 Fragmento de código de la implementación del modelado de tópicos (Parte 2)

4.6 Etapa 4: Presentación y discusión de resultados

En esta etapa, se evalúan los modelos o conjuntos de datos obtenidos, arrojados por las técnicas de minería de textos aplicadas en la anterior etapa. Dichos resultados son validados por medio de métricas automatizadas diseñadas para este fin; aunque también se miden con otros parámetros a tener en cuenta, como el tiempo de estimación y el tipo de respuesta a partir de lo obtenido. En el próximo capítulo se explica cómo se trabajó en función del cumplimiento de dicha etapa; de esa manera se dan respuestas a las tareas de investigación planteadas y se llegan a conclusiones.

Capítulo 5: Presentación y discusión de los resultados de la propuesta de solución

Introducción

En este capítulo se muestra cómo se ha trabajado para el cumplimiento de la "Etapa 4: Presentación y discusión de resultados" del "Procedimiento general en el análisis de artículos" establecido en el Capítulo 4 de la presente investigación. Se evalúan los modelos o conjuntos de datos obtenidos en la "Etapa 3: Análisis de la información", para proporcionar mejores resultados. Para ello se realiza un análisis y se arriban a conclusiones referente a la experimentación y evaluación de resultados por cada tarea trazada.

5.1 Selección de la métrica de evaluación para los algoritmos de clasificación de textos

Es de vital importancia establecer la métrica de evaluación adecuada porque esta regirá el buen desempeño de los algoritmos de clasificación de textos, en función de las tareas de investigación T1, T2 y T3, y permitirá llegar a las conclusiones generales y dar cumplimiento al objetivo general de la investigación.

En la sección "1.6 Métricas de evaluación en modelos predictivos" del Capítulo 1 de la presente investigación, se realiza una explicación de algunas de las métricas para la evaluación del rendimiento de un clasificador. En este sentido, la problemática en cuestión se enfocó en un problema de clasificación multi - clase, y cuando se analizó en la "Etapa 1: Recuperación de los documentos" del "Procedimiento general en el análisis de artículos" la cantidad de documentos que conforma el corpus para esta investigación; es de darse cuenta que no existe el mismo número de observaciones en cada clase (en función del problema: por cada área del conocimiento).

A partir de lo expuesto, no sería adecuado evaluar los clasificadores por la métrica de *accuracy*, sino por el puntaje de F1, donde se busca el equilibrio entre *precisión* (valor predictivo positivo) y *recall* (sensibilidad). A pesar que se ha decido tomar en cuenta los valores del puntaje F1, en las tablas de los resultados obtenidos por los clasificadores se muestran los resultados del puntaje para ambas métricas: *accuracy* y F1.

5.1.1 Presentación y discusión de resultados: tarea de investigación T1

Se muestra en la figura (Fig. 20) uno de los resultados obtenidos de la ejecución de los algoritmos de clasificación: *Multinomial Naïve Bayes, Logistic Regression* y Linear *Support Vector Classification* (Linear SVC), en función de la clasificación de los documentos a partir de todo su contenido.

```
It have read a total of:  272  files
Corpus:  (272, 272)
Train:  204 204
Test:  68 68
Train Stadistic:  {3.0: 73, 1.0: 43, 0.0: 41, 5.0: 18, 2.0: 7, 4.0: 22}
Test Stadistic:  {3.0: 29, 5.0: 4, 2.0: 2, 4.0: 8, 1.0: 14, 0.0: 11}
Reading file
 - Extracting features
 - Training Classifier
Reading Test files
 - Extracting Test features
 - Predicting Test
        Multinomial NB - F1_score 0.7403183885640027
        Logistic Regression - F1_score 0.8334472635732649
        Lineal SVC - F1_score 0.8302824053598048
```

Fig. 20 Resultados de la ejecución de los clasificadores (tarea de investigación T1)

En la tabla (Tabla 5) se muestra un resumen de los resultados de los experimentos realizados a partir de la variación de las características N-gramas para cada algoritmo de clasificación, mostrando el puntaje de la métrica F1.

Tabla 5 Resultados obtenidos de los clasificadores: tarea de investigación T1

N-gramas	**Variaciones**		
caracteres	{3-5}	{2-4}	{3-5}
palabras	{2-4}	{3-5}	{2-4}
etiquetas POS	{2-4}	{3-5}	{2-4}
saltos de palabras	{2-4}	{2-3}	{2-3}
palabras funcionales	{2-4}	{2-4}	{3-4}
símbolos de puntuación	{2-4}	{2-4}	{3-4}
Clasificador	**Métrica F1**		
Multinomial Naïve Bayes	0.741	0.7403	0.741
Logistic Regression	0.805	**0.833**	0.812
Linear SVC	0.792	0.8302	0.796

Se evidencia que el algoritmo: *Logistic Regression*, en este caso de la clasificación de los documentos a partir de todo su contenido, en conjunto con la combinación de los diferentes tipos de N-gramas (caracteres {2-4}, palabras {3-5}, etiquetas POS {3-5},

saltos de palabras {2-3}, palabras funcionales {2-4} y símbolos de puntuación {2-4}) muestra mejor rendimiento de F1 (0.833) que los otros algoritmos de aprendizaje supervisado y las diferentes variaciones de los N-gramas.

5.1.2 Presentación y discusión de resultados: tarea de investigación T2

Se muestra en la figura (Fig. 21) uno de los resultados obtenidos de la ejecución de los algoritmos de clasificación: *Multinomial Naïve Bayes, Logistic Regression* y Linear *Support Vector Classification* (Linear SVC), en función de la clasificación de los documentos a partir de su resumen.

```
It have read a total of:  272  files
Corpus:  (272, 272)
Train:  204 204
Test:  68 68
Train Stadistic:  {3.0: 73, 1.0: 43, 0.0: 41, 5.0: 18, 2.0: 7, 4.0: 22}
Test Stadistic:  {3.0: 29, 5.0: 4, 2.0: 2, 4.0: 8, 1.0: 14, 0.0: 11}
Reading file
 - Extracting features
 - Training Classifier
Reading Test files
 - Extracting Test features
 - Predicting Test
     Multinomial NB - F1_score 0.6414312617702448
     Logistic Regression - F1_score 0.5801469277019637
     Lineal SVC - F1_score 0.5725232621784345
```

Fig. 21 Resultados de la ejecución de los clasificadores (tarea de investigación T2)

En la tabla (Tabla 6) se muestra un resumen de los resultados de los experimentos realizados a partir de la variación de las características N-gramas para cada algoritmo de clasificación, mostrando el puntaje de la métrica F1.

Tabla 6 Resultados obtenidos de los clasificadores: tarea de investigación T2

N-gramas	**Variaciones**		
caracteres	{3-5}	{2-4}	{2-4}
palabras	{2-4}	{3-5}	{2-4}
etiquetas POS	{2-4}	{3-5}	{2-4}
saltos de palabras	{2-4}	{2-3}	{2-4}
palabras funcionales	{2-4}	{3-4}	{3-4}
símbolos de puntuación	{2-4}	{3-4}	{3-4}
Clasificador	**Métrica F1**		
Multinomial Naïve Bayes	0.636	0.636	**0.641**
Logistic Regression	0.585	0.581	0.580
Linear SVC	0.574	0.570	0.572

Se evidencia que el algoritmo: *Multinomial Naïve Bayes*, en este caso de la clasificación de los documentos a partir de su resumen, en conjunto con la combinación de los diferentes tipos de N-gramas (caracteres {2-4}, palabras {2-4}, etiquetas POS {2-4}, saltos de palabras {2-4}, palabras funcionales {3-4} y símbolos de puntuación {3-4}) muestra mejor rendimiento de F1 (0.641) que los otros algoritmos de aprendizaje supervisado y las diferentes variaciones de los N-gramas.

5.1.3 Presentación y discusión de resultados: tarea de investigación T3

Se muestra en la figura (Fig. 22) uno de los resultados obtenidos de la ejecución de los algoritmos de clasificación: *Multinomial Naïve Bayes, Logistic Regression* y Linear *Support Vector Classification* (Linear SVC), en función de la clasificación de los documentos a partir de sus palabras claves.

```
It have read a total of:  272  files
Corpus:  (272, 272)
Train:  204 204
Test:  68 68
Train Stadistic:  {3.0: 73, 1.0: 43, 0.0: 41, 5.0: 18, 2.0: 7, 4.0: 22}
Test Stadistic:  {3.0: 29, 5.0: 4, 2.0: 2, 4.0: 8, 1.0: 14, 0.0: 11}
Reading file
 - Extracting features
 - Training Classifier
Reading Test files
 - Extracting Test features
 - Predicting Test
     Multinomial NB - F1_score 0.5267200954455857
     Logistic Regression - F1_score 0.6133771929824561
     Lineal SVC - F1_score 0.4731837606837607
```

Fig. 22 Resultados de la ejecución de los clasificadores (tarea de investigación T3)

En la tabla (Tabla 7) se muestra un resumen de los resultados de los experimentos realizados a partir de la variación de las características N-gramas para cada algoritmo de clasificación, mostrando el puntaje de la métrica F1.

Tabla 7 Resultados obtenidos de los clasificadores: tarea de investigación T3

N-gramas	Variaciones		
caracteres	{2-4}	{2-4}	{1-3}
palabras	{2-3}	{1-2}	{1-2}
etiquetas POS	{2-3}	{1-2}	{1-2}
saltos de palabras	{1-2}	{1-2}	{1-2}
palabras funcionales	{1-2}	{2-3}	{1-2}
símbolos de puntuación	{1-2}	{1-2}	{1-2}

Clasificador	Métrica F1		
Multinomial Naïve Bayes	0.526	0.49	0.541
Logistic Regression	**0.613**	0.60	0.525
Linear SVC	0.473	0.51	0.527

Se evidencia que el algoritmo: *Logistic Regression*, en este caso de la clasificación de los documentos a partir de sus palabras claves, en conjunto con la combinación de los diferentes tipos de N-gramas (caracteres {1-3}, palabras {1-2}, etiquetas POS {1-2}, saltos de palabras {1-2}, palabras funcionales {1-2} y símbolos de puntuación {1-2}) muestra mejor rendimiento de F1 (0.613) que los otros algoritmos de aprendizaje supervisado y las diferentes variaciones de los N-gramas.

5.2 Presentación y discusión de resultados: tarea de investigación T4

Para poder identificar las palabras claves más frecuentes a lo largo de las diferentes áreas de conocimiento, se aplican los métodos de *stemmer*. Se muestra en la tabla (Tabla 8) los resultados obtenidos de las 10 palabras claves más comunes - más frecuentes - por cada área.

Tabla 8 Resultados obtenidos: tarea de investigación T4

Área: Inteligencia Artificial	
Método Porter: `[('robot', 24), ('system', 14), ('dialogu', 13), ('servic', 11), ('model', 11), ('speech', 10), ('of', 9), ('recognit', 8), ('cognit', 7), ('interact', 7)]` Tiempo transcurrido: 0.0179 s	*Método Porter2:* `[('robot', 24), ('system', 14), ('dialogu', 13), ('servic', 11), ('model', 11), ('speech', 10), ('of', 9), ('recognit', 8), ('cognit', 7), ('interact', 7)]` Tiempo transcurrido: 0.0149 s
Palabras claves frecuentes: robot, sistema, diálogo, servicio, modelo, habla, reconocimiento, cognición, interactuar	
Área: Redes y Seguridad en Cómputo	
Método Porter: `[('control', 34), ('system', 28), ('network', 24), ('distribut', 24), ('time', 18), ('frequenc', 13), ('fuzzi', 12), ('flow', 12), ('process', 10), ('doppler', 9)]`	*Método Porter2:* `[('control', 34), ('system', 28), ('network', 24), ('distribut', 24), ('time', 18), ('frequenc', 13), ('fuzzi', 12), ('flow', 12), ('process', 10), ('doppler', 9)]`

<table>
<tr><td>Tiempo transcurrido: 0.0189 s</td><td>Tiempo transcurrido: 0.0219 s</td></tr>
<tr><td colspan="2">Palabras claves frecuentes: control, sistema, red, distribuido, tiempo, frecuencia, fuzzi, flujo, proceso, doppler</td></tr>
<tr><td colspan="2">Área: Computación Científica</td></tr>
<tr><td>Método Porter:

<code>[('color', 7), ('graph', 3), ('polytop', 2), ('radon', 2), ('colloc', 2), ('method', 2), ('imag', 2), ('orient', 1), ('matroid', 1), ('cocircuit', 1)]</code>

Tiempo transcurrido: 0.0049 s</td><td>Método Porter2:

<code>[('ncolor', 5), ('graph', 2), ('nradon', 2), ('method', 2), ('color', 2), ('imag', 2), ('orient', 1), ('matroid', 1), ('ncocircuit', 1), ('ngraph', 1)]</code>

Tiempo transcurrido: 0.0039 s</td></tr>
<tr><td colspan="2">Palabras claves frecuentes: color, grafo, politopos, radón, colocación, método, imagen, orientación, matroides, cocircuito</td></tr>
<tr><td colspan="2">Área: Señales, Imágenes y Ambientes Virtuales</td></tr>
<tr><td>Método Porter:

<code>[('imag', 44), ('transform', 33), ('shape', 21), ('hermit', 21), ('segment', 19), ('filter', 18), ('discret', 15), ('code', 14), ('chain', 13), ('knot', 13)]</code>

Tiempo transcurrido: 0.0388 s</td><td>Método Porter2:

<code>[('imag', 44), ('transform', 33), ('shape', 21), ('hermit', 21), ('segment', 19), ('filter', 18), ('discret', 15), ('code', 14), ('chain', 13), ('knot', 13)]</code>

Tiempo transcurrido: 0.0359 s</td></tr>
<tr><td colspan="2">Palabras claves frecuentes: imagen, transformación, forma, hermita, segmentación, filtro, discreta, código, cadena, nudos</td></tr>
<tr><td colspan="2">Área: Ingeniería de Software y Bases de Datos</td></tr>
<tr><td>Método Porter:

<code>[('data', 31), ('qualiti', 14'), ('de', 7), ('agent', 7), ('softwar', 6), ('model', 6), ('integr', 6) ('mine', 6), ('engin', 5), ('essenc', 5)]</code>

Tiempo transcurrido: 0.0099 s</td><td>Método Porter2:

<code>[('data', 31), ('qualiti', 14'), ('de', 7), ('agent', 7), ('softwar', 6), ('model', 6), ('integr', 6) ('mine', 6), ('engin', 5), ('essenc', 5)]</code>

Tiempo transcurrido: 0.0099 s</td></tr>
<tr><td colspan="2">Palabras claves frecuentes: dato, calidad, agente, software, modelo, integración, minería, ingeniería, esencia</td></tr>
<tr><td colspan="2">Área: Teoría de la Computación</td></tr>
<tr><td>Método Porter:</td><td>Método Porter2:</td></tr>
</table>

`[('model', 9), ('design', 6), ('engin', 6), ('resolv', 5), ('search', 5), ('algorithm', 5), ('softwar', 5), ('backtrack', 4), ('optim', 4), ('driven', 4)]` Tiempo transcurrido: 0.0109 s	`[('model', 9), ('design', 6), ('engin', 6), ('resolv', 5), ('search', 5), ('algorithm', 5), ('softwar', 5), ('backtrack', 4), ('optim', 4), ('driven', 4)]` Tiempo transcurrido: 0.0099 s
Palabras claves frecuentes: modelo, diseño, ingeniería, resolución, búsqueda, algoritmo, software, backtrack, optimización, impulso	

A pesar de las ventajas y desventajas explicadas que tienen cada uno de estos métodos con respecto a cuándo es mejor utilizarlo - en cuestión del lenguaje - se pudo comprobar que para ambos métodos se obtuvo las mismas respuestas. Con respecto a su tiempo de ejecución, la diferencia son milésimas de segundos, lo cual se llega a la conclusión que no afectaría a la hora de decidir cuán mejor es un método respecto al otro.

5.3 Presentación y discusión de resultados: tarea de investigación T5

Luego de haberse construido diversos modelos LDA con todas las posibles combinaciones de los parámetros: número de tópicos y velocidad de aprendizaje; se obtiene el mejor modelo LDA a partir del mayor valor de *Log-likelihood* (explicado en la Etapa 3). Seguido el procedimiento, para el conjunto de documentos por cada una de las áreas de conocimiento, se ejecuta el método diseñado, y se muestra en un gráfico como se hace la selección del mejor modelo.

Posteriormente, se analizan los resultados a partir del mejor modelo obtenido, mostrándose la mezcla de tópicos y, por cada uno, una distribución de las 15 palabras principales que son representativas del tema. Por otra parte, la distribución de palabras por tópicos se visualiza haciendo uso de la biblioteca *PyLDAvis* (de la herramienta *Scikit-learn*). *PyLDAvis* ofrece la mejor visualización para ver la distribución de palabras clave y temas, para lo cual provee de una forma simple e interactiva analizar los resultados obtenidos. Esta visualización permite enriquecer lo desarrollado, teniéndose en cuenta las siguientes observaciones:

- Cada burbuja en la parte izquierda de la visualización representa un tópico. Cuanto más grande la burbuja, más predominante es ese tópico.
- Un buen modelo de tópicos es aquel que tiene burbujas bastante grandes y que no se solapan, dispersas en todo el gráfico en lugar de estar todas juntas y

clusterizadas en un único cuadrante. Un modelo con muchos tópicos seguramente tendrá burbujas pequeñas, ubicadas en una misma región del gráfico y con muchos casos de solapamiento

A continuación, se muestran los resultados obtenidos en función del cumplimiento de la tarea de investigación T5.

- Área: Inteligencia Artificial

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 2 tiene mejores puntajes, siendo de -106398.116 y, que la velocidad de aprendizaje (*Learning decay*) de 0.5 supera a 0.7 y 0.9; mostrados en la figura (Fig. 23).

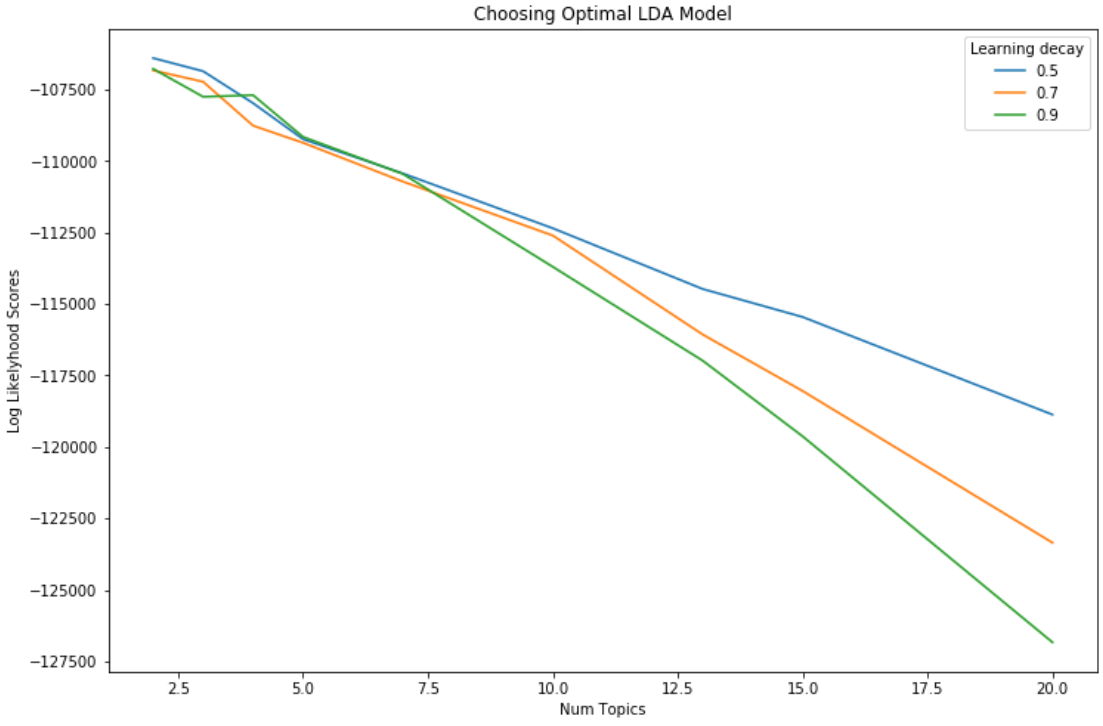

Fig. 23 Selección del Mejor Modelo LDA. Área: Inteligencia Artificial

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 2 temas. En la figura (Fig. 24) se puede observar la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

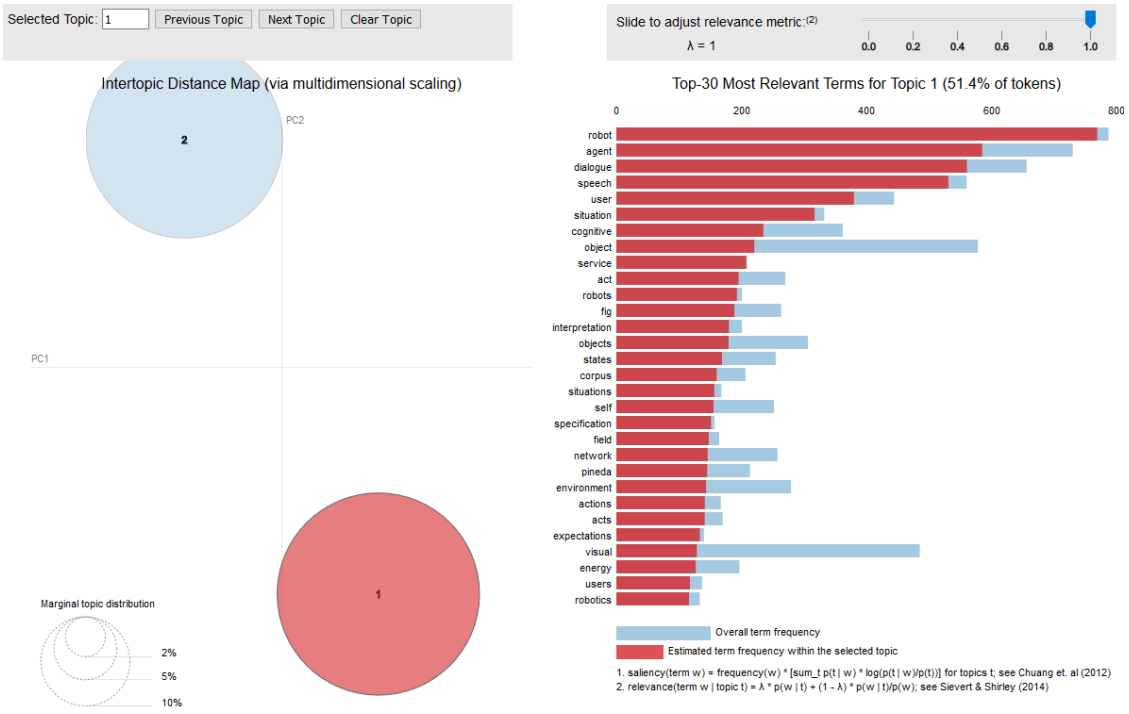

Fig. 24 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Inteligencia Artificial

Se muestra en el recuadro una distribución de las 15 palabras principales que son representativas del tema del total de 30 palabras que muestra el gráfico de la figura (Fig. 24). En este caso, la burbuja más grande es la 1, reflejándose en el *Topic #1* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de la Inteligencia Artificial compartan temas referentes a “servicios y tareas de robótica como el diálogo y servicios al usuario”.

Topic #0: object, visual, social, word, doi, words, image, neural, life, mind, images, agent, org, environment, classification

Topic #1: robot, agent, dialogue, speech, user, situation, cognitive, object, service, act, robots, fig, interpretation, objects, states

- Área: Redes y Seguridad en Cómputo

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 2 tiene mejores puntajes, siendo de -64326.108 y, que la velocidad de aprendizaje (*Learning decay*) de 0.9 supera a 0.5 y 0.7; mostrados en la figura (Fig. 25).

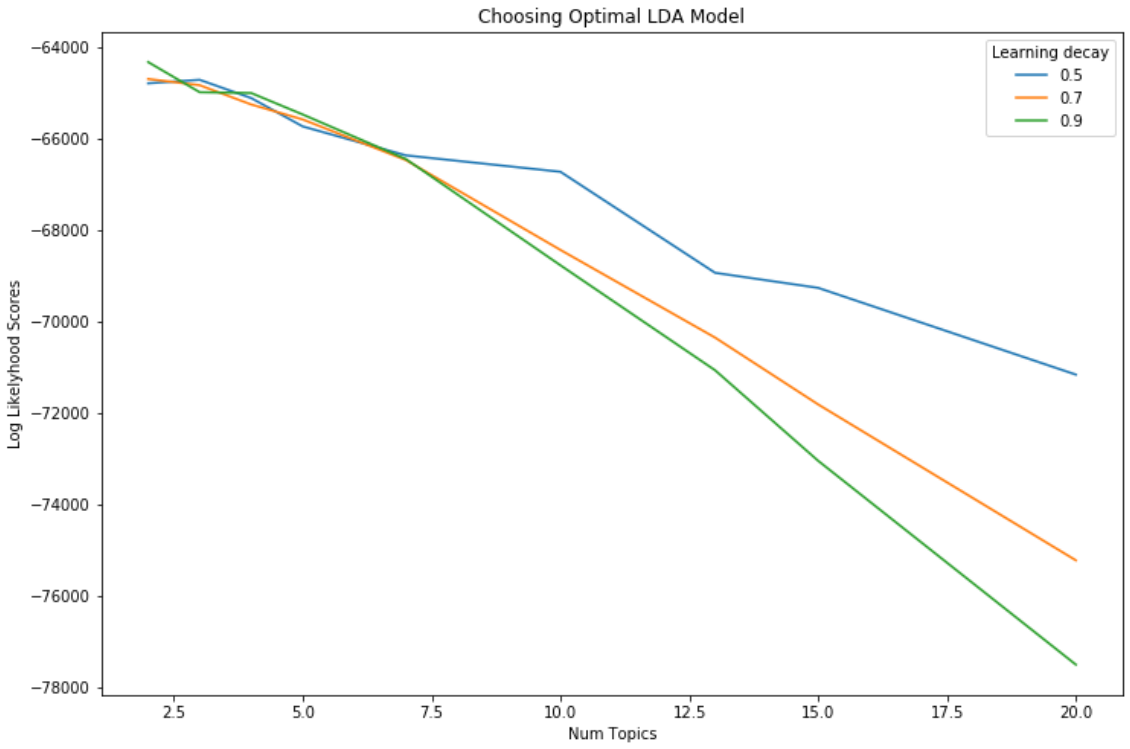

Fig. 25 Selección del Mejor Modelo LDA. Área: Redes y Seguridad en Cómputo

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 2 temas. Se puede observar en la figura (Fig. 26) la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

En el recuadro se muestra una distribución de las 15 palabras principales que son representativas del tema del total de 30 palabras que muestra el gráfico de la figura (Fig. 26). La burbuja más grande es la 1, reflejándose en el *Topic #0* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de Redes y Seguridad en Cómputo comparten temas referentes a “reconfiguración, flujo, distribución y escenarios discretos y difusos”.

Topic #0: fuzzy, delays, delay, flow, scenario, discrete, distribution, controller, law, scenarios, plant, neural, actuator, period, reconfiguration

Topic #1: nodes, processes, parallel, execution, transmission, sensors, frequencies, distance, scheduling, scheme, tasks, controller, region, type, level

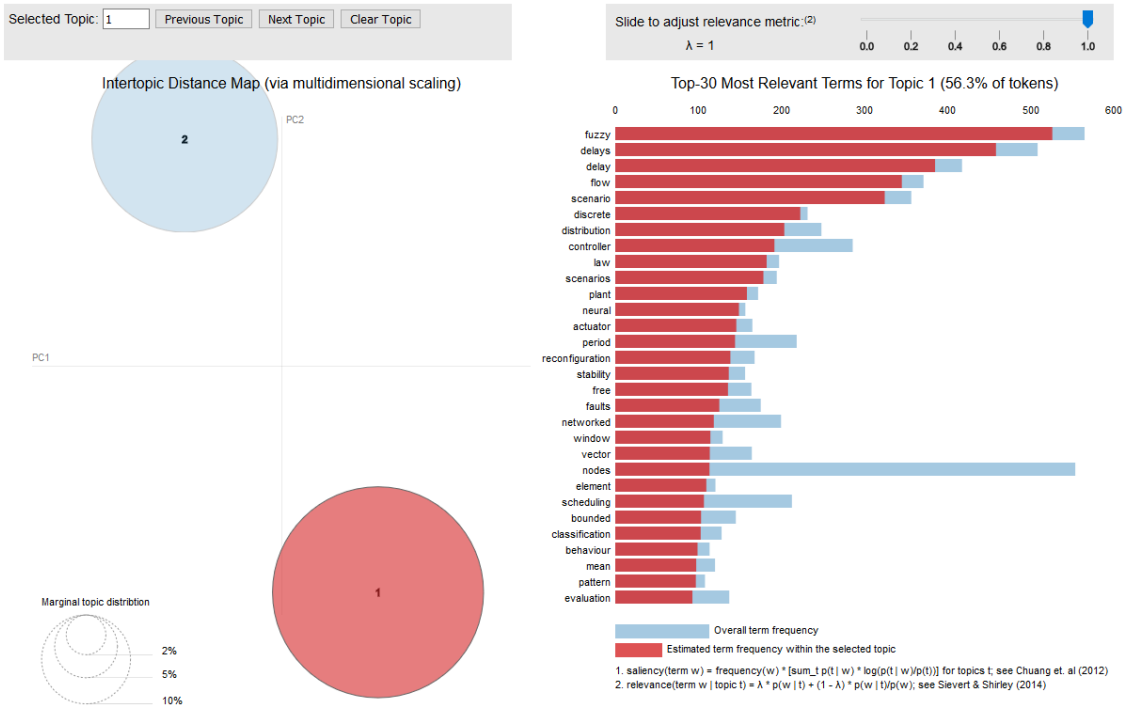

Fig. 26 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Redes y Seguridad en Cómputo

- Área: Computación Científica

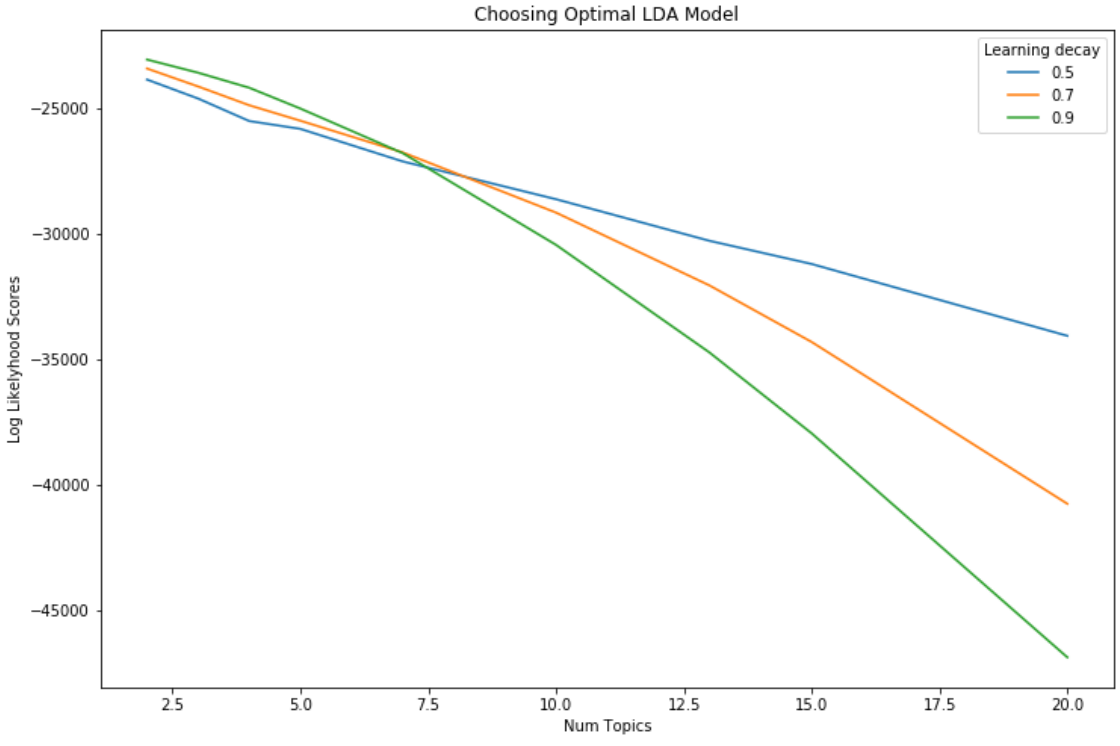

Fig. 27 Selección del Mejor Modelo LDA. Área: Computación Científica

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 2 tiene mejores puntajes, siendo de -23083.260 y, que la velocidad de aprendizaje (*Learning decay*) de 0.9 supera a 0.5 y 0.7; mostrados en la figura (Fig. 27).

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 2 temas. En la figura (Fig. 28) se puede observar la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

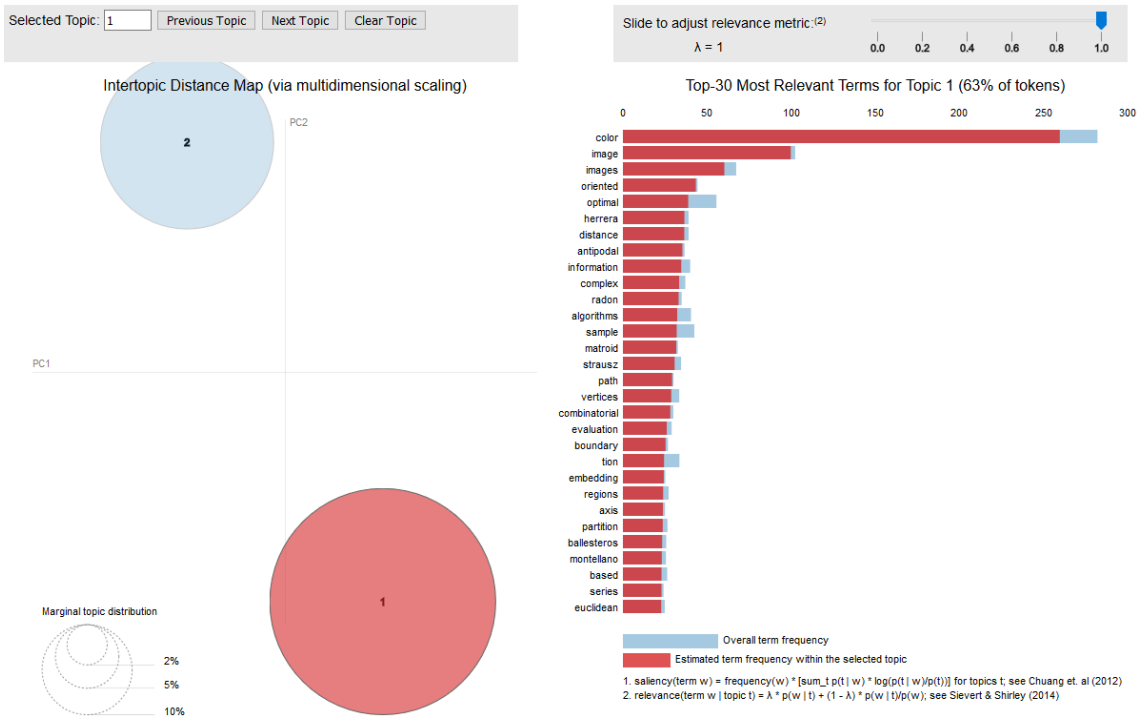

Fig. 28 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Computación Científica

Se muestra en el recuadro de un total de 30 palabras que muestra el gráfico de la figura (Fig. 28) una distribución de las 15 palabras principales que son representativas del tema. La burbuja más grande es la 1, reflejándose en el *Topic #0* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de Computación Científica

Topic #0: control, quantum, algebra, global, colors, phase, constrained, local, color, group, processes, clusters, complete, search, optimization

Topic #1: color, image, images, oriented, optimal, herrera, distance, antipodal, information, complex, radon, algorithms, sample, matroid, strausz

comparten temas referentes a “álgebra cuántica, procesos y fases como: búsqueda, restricción y optimización de clústeres”.

- Área: Señales, Imágenes y Ambientes Virtuales

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 2 tiene mejores puntajes, siendo de -107586.475 y, que la velocidad de aprendizaje (*Learning decay*) de 0.5 supera a 0.7 y 0.9; mostrados en la figura (Fig. 29).

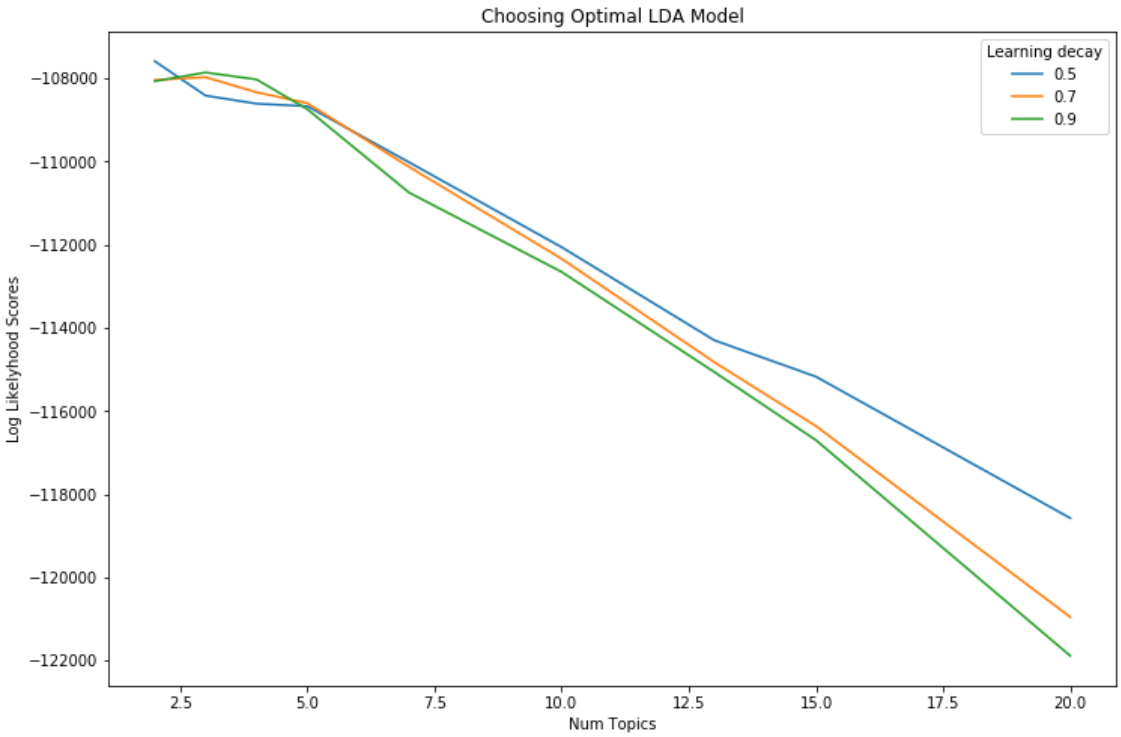

Fig. 29 Selección del Mejor Modelo LDA. Área: Señales, Imágenes y Ambientes Virtuales

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 2 temas. Se puede observar en la figura (Fig. 30) la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

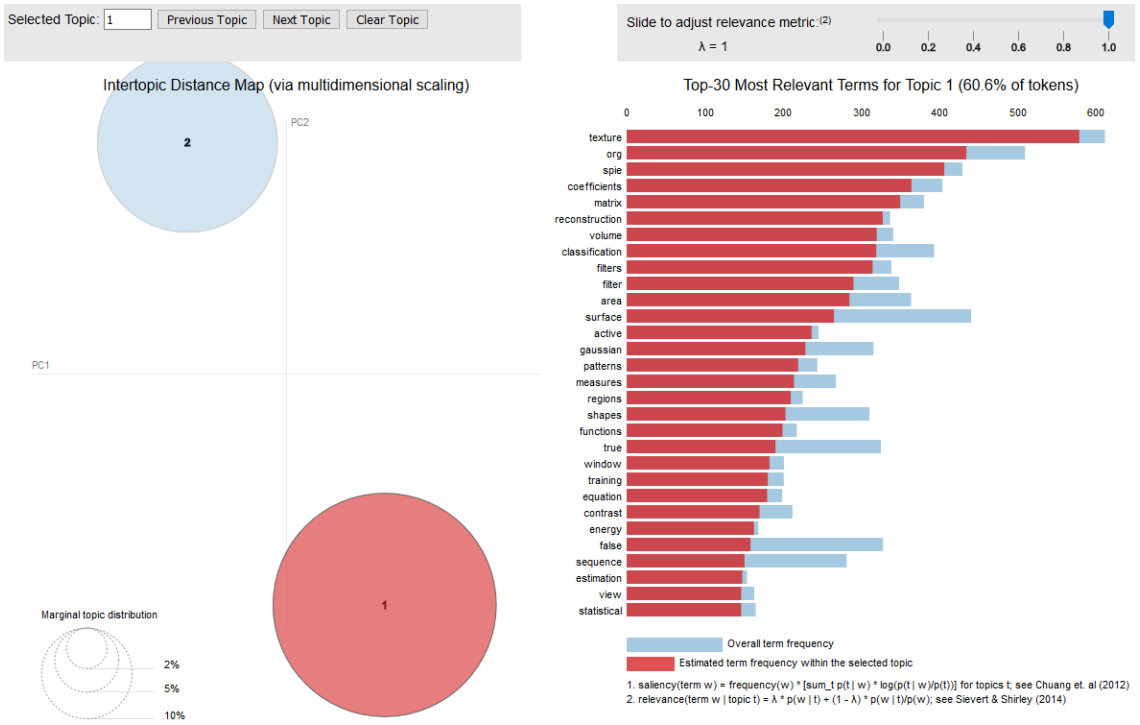

Fig. 30 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Señales, Imágenes y Ambientes Virtuales

En el recuadro se muestra de un total de 30 palabras que muestra el gráfico de la figura (Fig. 30) una distribución de las 15 palabras principales que son representativas del tema. La burbuja más grande es la 1, reflejándose en el *Topic #0* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de Señales, Imágenes y Ambientes Virtuales comparten temas referentes a "coeficientes de clasificación de textura, patrones y filtros, área, superficie y volumen gaussiano".

Topic #0: texture, org, spie, coefficients, matrix, reconstruction, volume, classification, filters, filter, area, surface, active, gaussian, patterns

Topic #1: chain, curve, color, descriptor, element, line, orthogonal, robust, robustness, plane, compression, equal, invariant, surface, false

- Área: Ingeniería de Software y Bases de Datos

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 3 tiene mejores puntajes, siendo de -45279.583 y, que la velocidad de aprendizaje (*Learning decay*) de 0.5 supera a 0.7 y 0.9; mostrados en la figura (Fig. 31).

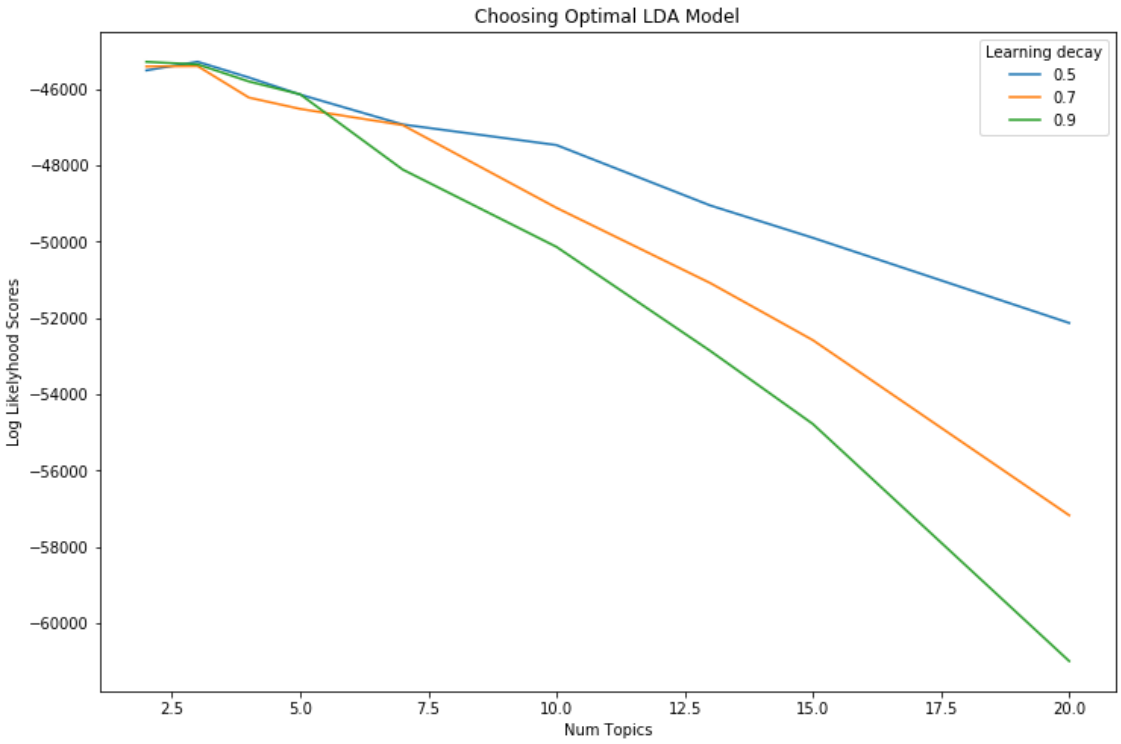

Fig. 31 Selección del Mejor Modelo LDA. Área: Ingeniería de Software y Base de Datos

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 3 temas. En la figura (Fig. 32) se puede observar la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

Se muestra en el recuadro una distribución de las 15 palabras principales que son representativas del tema de un total de 30 palabras que muestra el gráfico de la figura (Fig. 32). La burbuja más grande es la 1, reflejándose en el *Topic #1* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de Ingeniería de Software y Base de Datos comparten temas referentes a “registros, y algoritmos de minería, distancia y similitud”.

Topic #0: criteria, accuracy, provenance, scores, query, derived, resolution, granularity, consistency, measurement, manager, aggregation, metadata, priorities, primary

Topic #1: record, records, mining, algorithm, distance, matching, similarity, algorithms, practice, false, distributed, multi, scenario, precision, total

Topic #2: team, relational, organization, let, company, warehouse, query, alternative, configuration, states, space, members, study, Enterprise, false

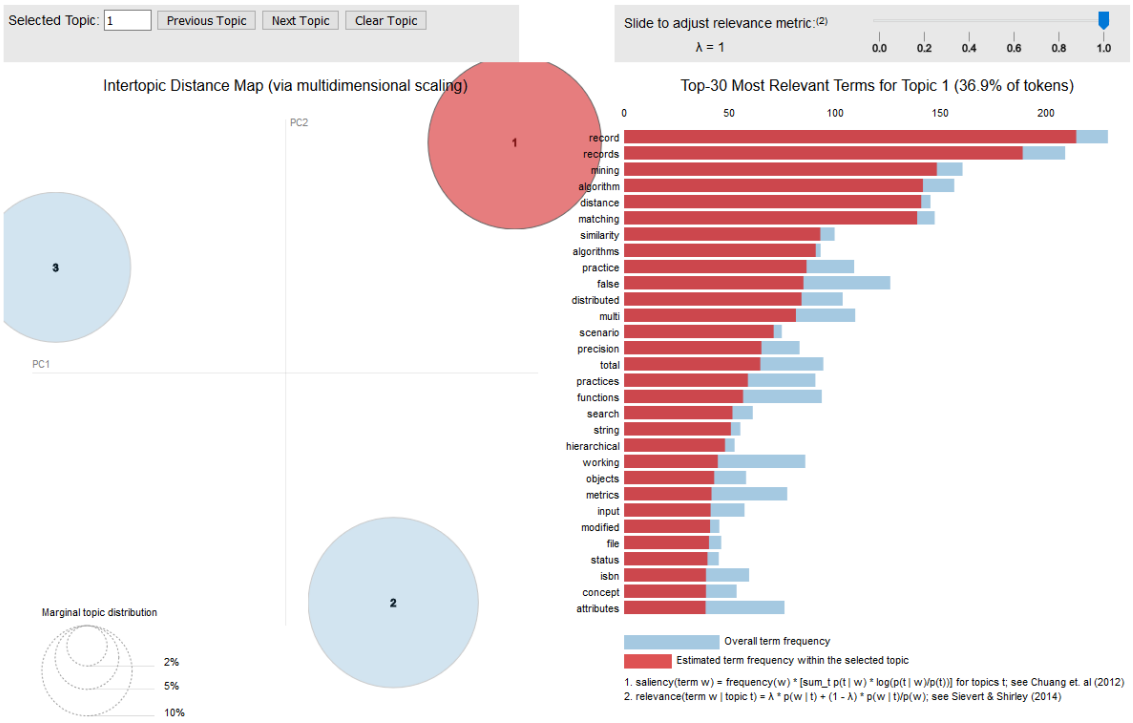

Fig. 32 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Ingeniería de Software y Base de Datos

- Área: Teoría de la Computación

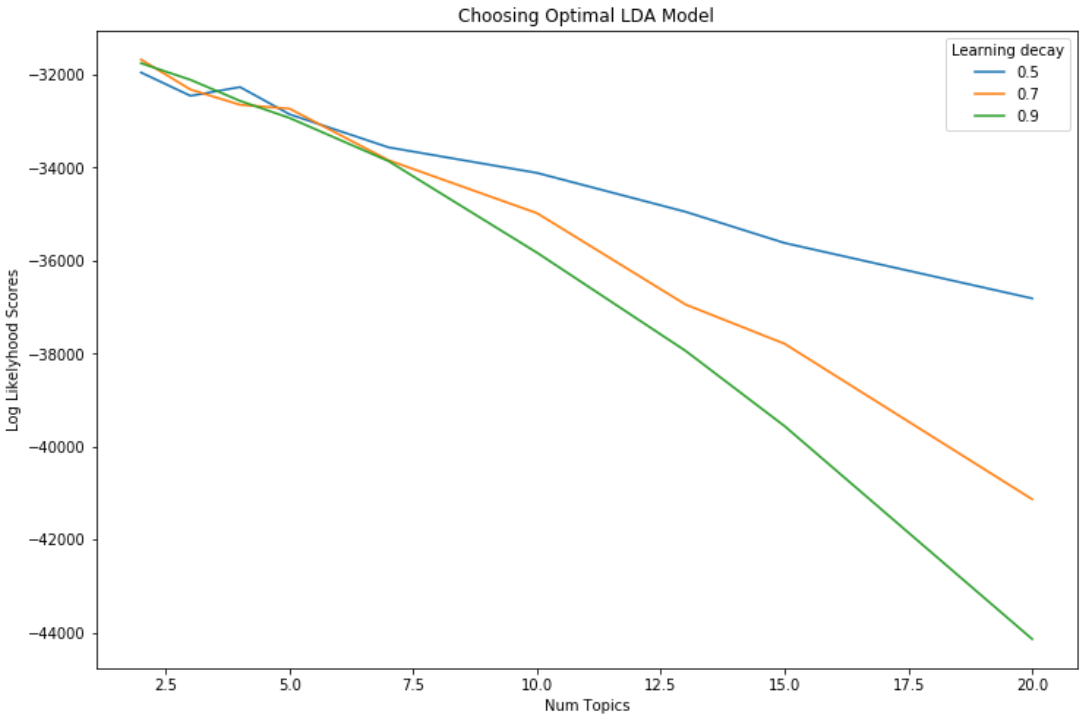

Fig. 33 Selección del Mejor Modelo LDA. Área: Teoría de la Computación

Al trazar los puntajes de probabilidad logarítmica (*Log Likelihood Scores*) contra número de tópicos (*Num Topics*), muestra que el número de temas igual a 2 tiene mejores puntajes, siendo de -31676.575 y, que la velocidad de aprendizaje (*Learning decay*) de 0.7 supera a 0.5 y 0.9; mostrados en la figura (Fig. 33).

Por lo tanto, una cantidad óptima de temas distintos para este conjunto de datos es de 2 temas. Se puede observar en la figura (Fig. 34) la distribución de palabras y temas del mejor modelo LDA de esta área, cumpliendo con las observaciones detectadas anteriormente respecto con este tipo de visualización.

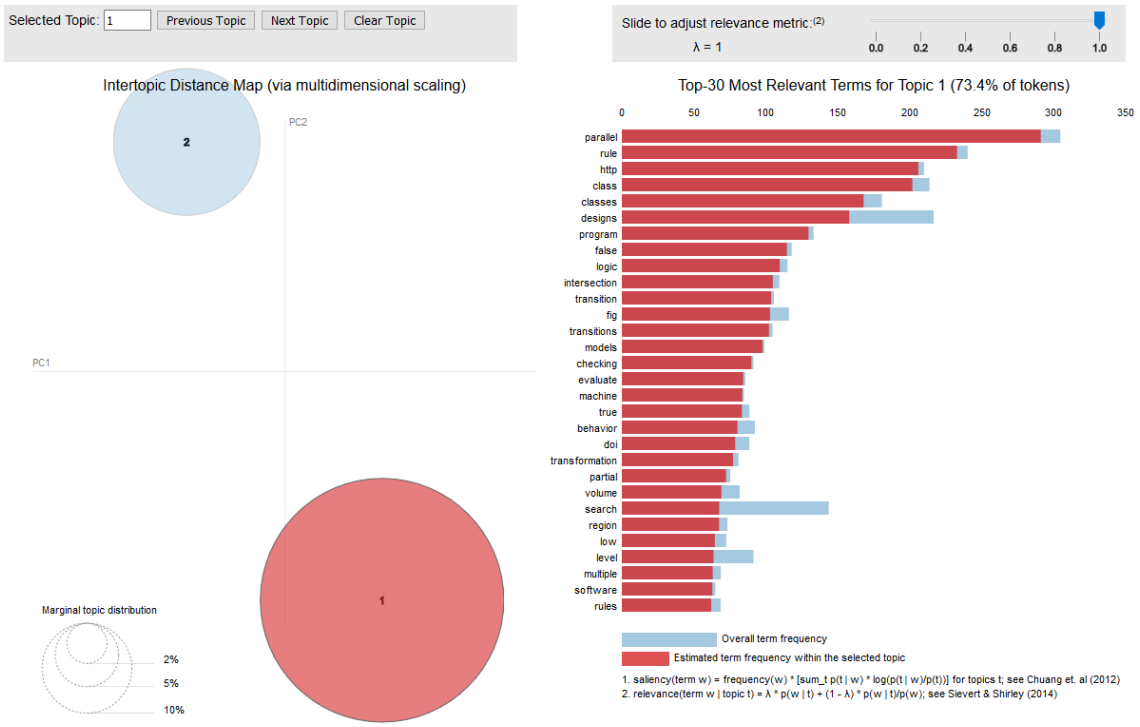

Fig. 34 Visualización de la distribución de palabras clave y temas del mejor modelo LDA. Área: Teoría de la Computación

En el recuadro se muestra de un total de 30 palabras que muestra el gráfico de la figura (Fig. 34) una distribución de las 15 palabras principales que son representativas del tema las 15 palabras claves por cada tópico.

Topic #0: parallel, rule, http, class, classes, designs, program, false, logic, intersection, transition, fig, transitions, models, checking

Topic #1: optimal, control, search, solutions, conditions, designs, face, average, optimization, mixed, journal, algorithms, functions, near, graph

La burbuja más grande es la 1, reflejándose en el *Topic #0* del recuadro, lo cual se puede inferir que del conjunto de documentos del área de Teoría de la Computación comparten temas referentes a "lógica e intersección de modelos, reglas, clases y diseños de programas".

5.4 Discusión en términos de palabras claves y temas compartidos

Como parte del planteamiento de las tareas de investigación, se realizó un análisis referente a cuáles palabras claves definidas por los autores de los artículos académicos eran comunes a lo largo de las diferentes áreas de conocimiento (tarea de investigación T4); así como identificar temas compartidos – comunes – que abarcan el conjunto de documentos por cada área de conocimiento (tarea de investigación T5). Para los dos casos en cuestión se aplicó un procedimiento general en función del análisis de los documentos, aplicándose algoritmos que solucionaron esta tarea, los cuales se optimizaron para la obtención de una mejor presentación en sus resultados.

Teniendo en cuenta los resultados obtenidos para ambos objetivos, en la sección "5.2 Presentación y discusión de resultados: tarea de investigación T4" del presente capítulo, se puede apreciar, que existen palabras claves muy comunes dentro de una misma área, así como en otras áreas. Por otro lado, en la sección "5.3 Presentación y discusión de resultados: tarea de investigación T5" del presente capítulo, se puede apreciar que los mejores modelos del modelado de tópico resultaron de 2 temas para cinco de las seis áreas del conocimiento planteadas y 3 temas para una de las seis áreas de conocimiento, donde se puede evidenciar la fuerte relación de los temas y su distribución de palabras que existen entre los documentos de una misma área.

A partir de este análisis, se puede concluir, que con la aplicación de estos algoritmos se pudieron detectar palabras y tópicos en común dentro de una misma área, así como en otras áreas del conocimiento, que pudieran sugerir algún tipo de redes de colaboración para la detección de aquellos artículos que tienen diferentes autores que tratan el mismo tema. Una red de colaboración tiene como propósito el estudio de temas disciplinares o multidisciplinares cuya relevancia amerite un esfuerzo colectivo. De esta manera, se pudieran responder a ciertas preguntas como, por ejemplo: ¿Cuáles son los tópicos que más le gusta escribir a ciertos autores? ¿Qué autores tienen afición por los mismos tópicos?

5.5 Discusión en términos del desempeño de los algoritmos de clasificación de textos

Como objetivo general de la presente investigación, se planteó desarrollar nuevas estrategias en la ejecución de algoritmos de minería de textos para un mejor resultado en el rendimiento del procesamiento de artículos científicos de investigadores en el área de las Ciencias de la Computación de la UNAM. Se trabajó en función del objetivo general, a partir de un análisis profundo en la manera de clasificar los documentos, tomándose en cuenta todo su contenido, solo su resumen o solo sus palabras claves, plasmado en las tareas de investigación.

Para los tres casos en cuestión, se aplicó un procedimiento general en función del análisis de los documentos para la obtención de un mejor rendimiento de los algoritmos de clasificación. De esa forma, se aplican varios pasos para el pre – procesamiento de los textos, así como la implementación de los clasificadores, en conjunto con la combinación de los diferentes tipos de N-gramas que fueron efectivos en la forma de alimentar a los clasificadores. En la tabla siguiente (Tabla 9) se puede observar un resumen de los resultados finales obtenidos en la manera de clasificar los documentos:

Tabla 9 Resultados de la clasificación de los documentos

Variaciones de N-gramas	Puntaje F1	Clasificador/tarea
caracteres {2-4}, palabras {3-5}, etiquetas POS {3-5}, saltos de palabras {2-3}, palabras funcionales {2-4} y símbolos de puntuación {2-4}	**0.833**	Logistic Regression (documento todo su contenido)
caracteres {2-4}, palabras {2-4}, etiquetas POS {2-4}, saltos de palabras {2-4}, palabras funcionales {3-4} y símbolos de puntuación {3-4}	0.641	Multinomial Naïve Bayes (documento solo su resumen)
caracteres {1-3}, palabras {1-2}, etiquetas POS {1-2}, saltos de palabras {1-2}, palabras funcionales {1-2} y símbolos de puntuación {1-2}	0.613	Logistic Regression (documento solo sus palabras claves)

Teniendo en cuenta estos resultados, se puede concluir, que el algoritmo de *Logistic Regression* presentó una mejor predicción respecto a los otros algoritmos, y que la mejor manera de clasificar los artículos científicos de investigadores en el área de las Ciencias de la Computación de la UNAM, es tomando todo su contenido y alimentando al clasificador con la variación de los diferentes tipos de N-gramas propuestos.

Conclusiones generales

Como resultado de la investigación se logró identificar y comparar el desempeño en el uso de los algoritmos de clasificación de textos para el análisis en artículos científicos de investigadores en el área de las Ciencias de la Computación de la Universidad Nacional Autónoma de México, por lo que se llegan a las siguientes conclusiones:

- El análisis de los diferentes trabajos académicos referentes a la minería de textos demostró que las soluciones no se ajustan a las necesidades y características de la problemática que se aborda en el presente trabajo, aunque sirvieron de fundamento en la presentación del procedimiento general en el análisis de los artículos desarrollado como propuesta de solución, así como las técnicas y herramientas de minería de textos para la clasificación de documentos.
- Con el trabajo, no solo se destaca la clasificación de documentos, sino que se reflejan otras técnicas, como el conteo de palabras comunes y la extracción de temas compartidos para un conjunto de documentos clasificados por áreas de conocimiento; abarcando las técnicas de minería de textos: el aprendizaje supervisado y el aprendizaje no supervisado.
- Se comprueba que con la aplicación de varios pasos para el pre – procesamiento de los artículos científicos, tomando todo su contenido y alimentando al clasificador con la variación de los diferentes tipos de N-gramas (caracteres {2-4}, palabras {3-5}, etiquetas POS {3-5}, saltos de palabras {2-3}, palabras funcionales {2-4} y símbolos de puntuación {2-4}) presenta el algoritmo de *Logistic Regression* una mejor predicción de 0.833 respecto a los otros algoritmos implementados: *Multinomial Naïve Bayes* y *Linear Support Vector Classification*, y las diferentes variaciones de los N-gramas; permitiendo a los usuarios académicos realizar análisis de textos usando esta propuesta para encontrar artículos relevantes en su área de interés.

Trabajo a futuro

La presente investigación abarcó el análisis de artículos científicos de las Ciencias de la Computación de la UNAM, específicamente, de investigadores del Posgrado de Ciencias e Ingeniería de la Computación, y la clasificación de los artículos a partir de las áreas del conocimiento establecidas en el posgrado, por lo que se recomienda para trabajo futuro:

- Trabajar la clasificación de los documentos desde una perspectiva multi – clase y multi – etiqueta, y comparar el desempeño y rendimiento de los algoritmos. Debido a que, en la etapa de etiquetado de los documentos, en la que se tuvo en cuenta el criterio de tres anotadores, se presentaron casos de artículos que pudieron ser etiquetado con más de una categoría de clasificación. En este sentido, los algoritmos de clasificación de textos, están basado en ocurrencias de palabras, por lo que pudiera existir la posibilidad que seleccione una categoría diferente a la establecida de antemano que, de alguna manera, ambas clasificaciones no estarían mal.
- Analizar y extraer más información de los artículos, pero de una misma revista de publicación. Se debe a que fueron artículos extraídos de diferentes lugares de publicación, lo cual se tuvo que realizar un engorroso trabajo en la etapa de pre – procesamiento, debido a la estructura del formato en los mismos y a su vez, la diversidad de patrones que presentaron todos ellos, por ejemplo, en el momento de extracción del *abstract* y las *keywords*, lo cual imposibilitó extraer y analizar más información como, por ejemplo: tipo de métodos o algoritmos por áreas de conocimiento que son más utilizados.
- Sugerir un tipo de redes de colaboración para la detección de aquellos artículos que tienen diferentes autores que tratan el mismo tema. Esto se debe, a partir del análisis realizado con la frecuencia de palabras claves y temas compartidos a lo largo de una misma área de conocimiento, donde con la aplicación de estos algoritmos se pudieron detectar palabras y tópicos en común dentro de una misma área, así como en otras áreas del conocimiento.

Referencias Bibliográficas

Abbott, D. (2013). Introduction to Text Mining. *Virtual Data Intensive Summer School.* Obtenido de http://www.vscse.org/summerschool/2013/Abbott.pdf

Aggarwal, C. C., & Zhai, C. (2012). *Mining text data.* Springer Science & Business Media.

Altman, D. (1991). *Practical statistics for medical research.* London: Chapman and Hall.

AMPLN. (2019). *CICLing: International Conference on Computational Linguistics and Intelligent Text Processing*. Obtenido de AMPLN Asociación Mexicana para el Procesamiento del Lenguaje: https://www.cicling.org/ampln/

Ángeles L., M. I., & Santillán G., A. M. (2001). Minería de datos: concepto, características, estructura y aplicaciones.

Arias C., A., Mattos S., Y., Heredia G., J., & Heredia V., D. (2016). Minería de texto como una herramienta para la búsqueda de artículos científicos para la investigación. *Revista I+D en TIC , 7*(1), 14-20. Obtenido de http://publicaciones.unisimonbolivar.edu.co/rdigital/ojs/index.php/identic/index

Beitzel., S. M. (2006). *On Understanding and Classifying Web Queries (Ph.D. thesis).*

Bijalwan, V., Kumar, V., Kumari, P., & Pascual, J. (2014). KNN based Machine Learning Approach for Text and Document Mining. *International Journal of Database Theory and Application, 7*(1), 61-70. Obtenido de http://web.inf.ufpr.br/menotti/ci171-2015-2-1/files/seminario-Fabricio-artigo.pdf

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research, 3(Jan)*, 993-1022.

Brownlee, J. (2016). *Metrics To Evaluate Machine Learning Algorithms in Python*. Obtenido de Machine Learning Mastery: https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/

Cohen, j. (1960). *A coefficient of agreement for nominal scales.* Educational and Psychological Measurement, 20:37-46.

Evaluation Measures for Data Mining Tasks. (2014). Obtenido de Winter School on "Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets": http://iasri.res.in/ebook/win_school_aa/notes/Evaluation_Measures.pdf

FreeLing. (s.f.). *FreeLing Home Page*. Obtenido de http://nlp.lsi.upc.edu/freeling/

Gómez G., J. (1998). Fundamentos de Lingüistica Computacional: bases teóricas, líneas de investigación y aplicaciones. *Revistes Catalanes amb Accés Obert (RACO)*, 135-146. Obtenido de https://www.raco.cat/index.php/Bibliodoc/article/viewFile/56629/66051

Gomez-Adorno, H., Bel-Enguix, G., Sierra, G., Sánchez, O., & Quezada, D. (2018). A machine learning approach for detecting aggressive tweets in spanish. *In Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), CEUR WS Proceeding.*

Gónzalez G., C. E., Montes, A., Sierra, G., Núñez J., J. A., Salinas L., A. J., & Ek, J. (2015). Tweets Classification Using Corpus Dependent Tags, Character and POS N-grams. *Notebook for PAN at CLEF, 1391*. Obtenido de http://ceur-ws.org/Vol-1391/104-CR.pdf

Gösling, J., & J., D. (2009). Information Retrieval: Searching in the 21st Century. Reino Unido: John Wiley and Sons, Ltd.

Greene, W. H. (2012). *Econometric Analysis (Seventh ed.).* Boston: Pearson Education.

Hearst. (1999). Untangling Text Data Mining. *Proc. of ACL'99: The 37th Annual Meeting of the Association for Computational Linguistics.* University of Maryland.

Jupyter, T. (2015). *Documentation The Jupyter Notebook*. Obtenido de https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

Justicia de la T., M. d. (2017). Nuevas Técnicas de Minería de Textos: Aplicaciones. *Universidad de Granada. Tesis Doctorales*. Obtenido de http://hdl.handle.net/10481/46975

Knerr, S., Personnaz, L., & Dreyfus, G. (1990). Single-layer learning revisited: A stepwise procedure for building and training neural network. *Neurocomputing: Algorithms, Architectures and Applications, NATO ASI, Berlin: Springer-Verlag*.

Kodratoff. (1999). Knowledge Discovery in Texts: A Definition and Applications. *Proc. of the 11th International Symposium on Foundations of Intelligent Systems (ISMIS-99).*

Kulmizev, A., Blankers, B., Bjerva, J., Nissim, M., van Noord, G., Plank, B., & Wieling, M. (2017). The power of character n-grams in native language identification. *In Proceedings of the 12th Workshop on Innovative Use of NLP*, (págs. 382-389).

Li, Y.-Y. W., & Acero, A. (2008). Learning query intent from regularized click graphs. *Proceedings of the 31st SIGIR Conference.*

Lobos, M. E. (2005). *Aprende a programar. Capítulo 4: Concepto de lenguaje de programación.*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781.

Montes-y-Gómez, M. (2001). Minería de texto: Un nuevo reto computacional. Obtenido de https://ccc.inaoep.mx/~mmontesg/publicaciones/2001/MineriaTexto-md01.pdf

NLTK, P. (s.f.). *Natural Language Toolkit - NLTK 3.4 documentation*. Obtenido de http://www.nltk.org/

NumPy. (2019). *NumPy*. Obtenido de https://www.numpy.org/

pandas. (2019). *pandas: Python Data Analysis Library*. Obtenido de https://pandas.pydata.org/

PorterStemmer. (2006). *The Porter Stemming Algorithm*. Obtenido de https://tartarus.org/martin/PorterStemmer/

Python. (s.f.). *Python TM*. Obtenido de https://www.python.org/

Riquelme, J. C., Ruíz, R., & Gilbert, K. (2006). Minería de datos: conceptos y tendencias. *Inteligencia Artificial,, 10*(29).

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of information by computer. *E.U.A: Eddision Wesley*.

Sánchez J., J., Villaseñor P., L., Escalante, H. J., & Montes y G., M. (2007). Detección del engaño en notas de opinión a través de técnicas tradicionales de clasificación automática de textos. *Noveno Congreso Mexicano de Inteligencia Artificial (COMIA 2017). Toluca, Mexico, 134*, 141-150. Obtenido de http://www.rcs.cic.ipn.mx/rcs/2017_134/Deteccion%20del%20engano%20en%20notas%20de%20opinion%20a%20traves%20de%20tecnicas%20tradicionales%20de%20clasificacion.pdf

Sánchez, D., & Martín-Bautista, M. (2008). Un enfoque deductivo para la minería de texto. Obtenido de http://www.softcomputing.es/estylf08/es/2006-XIII%20Congreso/articulos/40.pdf

Sanchez-Perez, M., Markov, I., Gómez-Adorno, H., & Sidorov, G. (2017). Comparison of character n-grams and lexical features on author, gender, and language variety identification on the same spanish news corpus. *In International Conference of the Cross-Language Evaluation Forum for European Languagues* (págs. 145-151). Springer.

scikit-learn. (s.f.). *1.1.11. Logistic regression*. Obtenido de https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

scikit-learn. (s.f.). *1.10. Decision Trees*. Obtenido de scikit-learn: Machine Learning in Python: http://scikit-learn.org/stable/modules/tree.html

scikit-learn. (s.f.). *1.4. Support Vector Machines*. Obtenido de scikit-learn: Machine Learning in Python: http://scikit-learn.org/stable/modules/svm.html

scikit-learn. (s.f.). *1.9. Naive Bayes*. Obtenido de scikit-learn: Machine Learning in Python: http://scikit-learn.org/stable/modules/naive_bayes.html

scikit-learn. (s.f.). *2.3. Clustering*. Obtenido de scikit-learn: Machine Learning in Python: http://scikit-learn.org/stable/modules/clustering.html

scikit-learn. (s.f.). *scikit-learn: Machine Learning in Python*. Obtenido de https://scikit-learn.org/stable/index.html

Snowball. (2006). *Snowball*. Obtenido de http://snowballstem.org/

spacy. (2019). *spacy: Industrial-Strength Natural Language Processing*. Obtenido de https://spacy.io/

Sukanya M., S. (2012). Techniques on Text Mining. *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, (págs. 269-271). Ramanathapuram.

Tan. (1999). Text Mining: The state of the art and challenges. *Proc. of the Workshop Knowledge Discovery from advanced Databases PAKDDD-99*.

Tao, J., Zheng, F., Li, A., & Li, Y. (2009). Advances in Chinese Natural Language Processing. *Speech Database and Assessments, 2009 Oriental COCOSDA Internacional COnference on* (págs. 13-18). IEEE Conference Publications.

Villatoro T., E., Anguiano, E., Montes y G., M., Villaseñor P., L., & Ramírez de la Rosa, G. (2016). Enhancing Semi-Supervised Text Classification using Document Summaries. *Ibero-American Conference on Artificial Intelligence. IBERAMIA 2016: Advances in Artificial Intelligence*, 115-126. Obtenido de https://ccc.inaoep.mx/~mmontesg/publicaciones/2016/SummariesForSemisupervisedTextClassification-IBERAMIA16.pdf

Weiss, S., Indurkhya, N., Zhang, T., & Damerau, F. (2005). Text Mining, Predictive Methods for Analyzing Unstructured Information. *Nueva York: Springer Science*.

Xu, F., Kurz, D., & Piskorski, J. (2002). *Term extraction and mining of term relations from unrestricted texts in the financial domain.* In Proceedings of BIS.

Zhang, H. (2004). *The Optimality of Naïve Bayes.*

Zhao, Y., & Karypis, G. (2005). *Topic-driven clustering for document datasets.* In SDM.