UNIVERSIDAD
DE GRANADA

Facultad de Ciencias. Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación

Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Minería de bases de conocimiento

Presentado por:
Mónica Calzado Granados

Curso académico 2023-2024

Minería de bases de conocimiento

Mónica Calzado Granados

Mónica Calzado Granados *Minería de bases de conocimiento*.
Trabajo de fin de Grado. Curso académico 2023-2024.

Responsable de tutorización

Úrsula Torres Parejo
Departamento de Estadística e Investigación Operativa

Daniel Sánchez Fernández
Departamento de Ciencias de la Computación e Inteligencia Artificial

Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias.
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Universidad de Granada

Declaración de originalidad

D./Dña. Mónica Calzado Granados

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 12 de marzo de 2024

Fdo: Mónica Calzado Granados

Índice general

Agradecimientos

Agradecimientos (opcional, ver archivo `preliminares/agradecimiento.tex`).

Summary

An english summary of the project (around 800 and 1500 words are recommended).
File: `preliminares/summary.tex`

Introducción

De acuerdo con la comisión de grado, el TFG debe incluir una introducción en la que se describan claramente los objetivos previstos inicialmente en la propuesta de TFG, indicando si han sido o no alcanzados, los antecedentes importantes para el desarrollo, los resultados obtenidos, en su caso y las principales fuentes consultadas.

Ver archivo `preliminares/introduccion.tex`

1. Minería de textos

La Minería de textos es una rama de estudio crucial en la Ciencia de Datos. Se ha convertido en una herramienta invaluable para extraer conocimiento a partir de grandes cantidades de texto no estructurado.

En los últimos años, la Minería de textos ha ganado una gran importancia debido al explosivo crecimiento de datos textuales que se pueden encontrar en diversas fuentes: documentos, correos electrónicos, redes sociales, páginas web y otros tipos de contenido textual.

Esta disciplina se ha vuelto esencial en medida que se encarga de analizar y descubrir nuevo patrones, tendencias y relaciones significativas dentro de conjuntos de información no estrucutrada. Mediante el uso de técnicas avanzadas de procesamiento del lenguaje natural (PLN) y aprendizaje automático, la minería de textos permite a las empresas, científicos y organizaciones obtener información valiosa que les ayude a impulsar una toma de decisiones estratégica.

1.1. Knowledge Discovery from Databases (KDD)

La información contenida en las bases de datos puede ser analizada de forma manual por expertos para obtener conclusiones. Sin embargo, en estos tiempos de avance tecnológico, debido al gran volumen de datos, necesitamos encontrar una forma de automatizar el proceso.

Surge así el Descubrimiento de Conocimiento en Bases de Datos (en inglés KDD, Knowledge Discovery from Databases), que tiene sus bases en disciplinas como la Estadística, Sistemas de Información y Aprendizaje Automático. El término KDD se puede definir como:

> *El proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos* [1].

1.1.1. Fases del KDD

El proceso de KDD resulta bastante interactivo, en el sentido de que intervienen muchas decisiones tomadas por el usuario según este considere oportuno. Estos pasos se pueden resumir de la siguiente manera [1]:

1. **Aprendizaje sobre el dominio de la aplicación:** es necesario determinar los objetivos y reunir el conocimiento previo que tenemos sobre el problema.

2. **Creación de un corpus de datos:** seleccionamos un dataset del cual queremos extraer conocimiento.

3. **Limpieza de datos y preprocesamiento:** partiendo de los datos recopilados, esta es la fase en la que se eliminan los datos ruidosos y anómalos. También se llevan a cabo estrategias para manejar valores perdidos.

4. **Reducción de datos:** consiste en seleccionar las características más útiles para representar los datos, y reducir la dimensionalidad del problema descartar atributos poco relevantes.

5. **Elegir una técnica de minería de datos:** se trata de decidir el propósito del modelo (qué tipo de información se quiere descubrir) y qué técnica de minería de datos podemos aplicar para ello (sumarización, clasificación, regresión, clustering, etc).

6. **Elegir un algoritmo de minería de datos:** incluye decidir qué modelos y parámetros pueden ser apropiados (por ejemplo, existen modelos para datos categóricos que son distintos de los modelos para datos numéricos).

7. **Minería de datos:** consiste en el descubrimiento de patrones, relaciones y conocimiento de interés, mediante el uso de algoritmos de clustering, clasificación, regresión, asociación, y otras técnicas de aprendizaje automático.

8. **Interpretación:** evaluar los patrones descubiertos y regresar a cualquiera de los pasos anteriores si es necesario. También trata la representación y visualización de los patrones extraídos, para facilitar la comprensión por los usuarios.

9. **Utilización del conocimiento descubierto:** consiste en acciones desde incorporar el nuevo conocimiento en un sistema inteligente, llevar a cabo acciones respaldadas en dicho conocimiento, o simplemente documentarlo y contrastarlo con conocimiento anteriormente extraído.

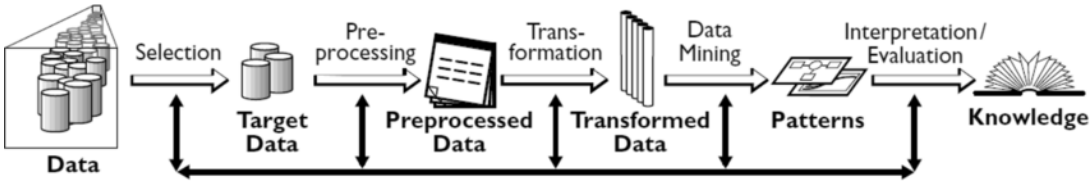

Figura 1.1.: Fases del proceso KDD [1]

La mayoría de los estudios sobre KDD se enfocan principalmente en el paso de la minería de datos. Sin embargo, los otros pasos son igualmente importantes para la aplicación del proceso de KDD.

1.2. Knowledge Discovery from Text (KDT)

La mayoría de información generada por las organizaciones se encuentra en forma de texto. Procesar dicha información puede resultar difícil debido al gran volumen de documentos y la falta de estructura de estos con respecto a una base de datos convencional. Es por ello que necesitaremos tratar algunas de las fases del KDD de forma distinta al tratar con textos.

1.2.1. KDD vs KDT

En general, la diferencia principal entre KDD y KDT reside en que el texto carece de una estructura procesable de forma automática, por lo que las etapas de selección, preprocesamiento

y transformación de los datos textuales se convierten en fundamentales e imprescindibles [2]. A partir de ahora nos referiremos a esta etapa como Preprocesamiento. [Puedo insertar tabla de comparación]

1.2.2. Fases del KDT

Podemos dividir las principales fases del KDT en tres [2]:

- **Preprocesamiento:** antes de realizar la minería, necesitamos preparar el texto a un formato que sea aceptado por las técnicas que queremos aplicar. Esto abarca desde la homegenización de distintos documentos, hasta la representación en Formas Intermedias. Entre las distintas técnicas de transformación de texto encontramos las siguientes:
  1. Normalización de texto: convertir todo el texto a minúsculas o mayúsculas para que las palabras sean tratadas de manera uniforme.
  2. Tokenización: dividir el texto en unidades más pequeñas, como palabras o frases, llamadas tokens.
  3. Etiquetado: asignar a cada token una etiqueta que indica su función gramatical: sustantivo, verbo, adjetivo, etc.
  4. Eliminación de *stop words*: eliminar palabras comunes pero no informativas que no aportan significado al texto, como determinantes, preposiciones, etc.
  5. *Stemming*: Reducir las palabras a su raíz, eliminando sufijos y prefijos, para reducir las palabras a su forma base.
  6. Corrección ortográfica: Corregir errores ortográficos comunes para mejorar la precisión del análisis.
  7. Eliminación de palabras raras o infrecuentes: Eliminar palabras que aparecen con poca frecuencia en el corpus, ya que pueden no ser útiles para el análisis.
  8. Eliminación de URLs, referencias y etiquetas HTML, que no aportan información semántica.
  9. Uso de *synsets* [3] que identifican conceptos a partir de palabras sinónimas, haciendo uso de diccionarios ontológicos.
- **Minería de textos**: En la literatura, algunos autores identifican el término *Minería de Textos* con todo el proceso de Descubrimiento de Conocimiento en Texto (KDT) [2]; sin embargo, en este contexto, nos referiremos específicamente a la fase encargada de localizar patrones, relaciones y estructuras interesantes, útiles y desconocidos en el texto.
- **Visualización**: una vez obtenidos los resultados de la fase anterior, es necesaria una fase de visualización para representar de manera efectiva los patrones, términos, tendencias y los resultados descubiertos. La visualización juega un papel crucial en la comprensión de la información extraída, permitiendo a los usuarios explorar y analizar los datos de manera intuitiva y amigable.
  Se pueden emplear numerosas herramientas dependiendo de la técnica de minería empleada, como nubes de palabras, diagramas de ocurrencia de términos, gráficos de dispersión de términos, clusterización de documentos, etc. Estas herramientas no sólo facilitan la interpretación de los resultados, sino que también pueden ayudar a descubrir más patrones, relaciones semánticas entre términos, etc.

1.2.3. Aplicaciones del KDT

Debido a la dificultad inherente de estructurar y procesar el texto de manera adecuada, las aplicaciones de la Minería de Texto enfrentan desafíos bastante complejos. Al igual que en la Minería de Datos, los expertos humanos tienen un papel crucial a la hora de interpretar los resultados de manera efectiva. Entre las distintas aplicaciones del KDT destacamos las siguientes [2]:

1. **Segmentación de clientes:** se trata de clasificar a los clientes en grupos según patrones de compra, preferencias o características demográficas. Esta segmentación permite a las empresas satisfacer mejor las necesidades del cliente y realizar estrategias de Marketing más eficaces.

2. **Detección de fraude financiero:** este es un desafío constante en el sector financiero debido a la evolución de las técnicas empleadas por los delincuentes. Existen algoritmos de detección de anomalías, que se utilizan para detectar transacciones y patrones de compras inusuales para un cliente específico.

3. **Filtrado de correos electrónicos:** este problema surge debido al volumen considerable de mensajes que un usuario puede recibir diariamente, lo que puede resultar abrumador sin herramientas para gestionarlos. La implementación de filtros de correo electrónico puede mejorar significativamente la productividad en el lugar de trabajo, al ayudar a procesar los correos de forma más eficiente.

4. **Análisis de sentimientos:** se trata de una aplicación crucial para comprender la opinión y el comportamiento de los usuarios en plataformas como Twitter, Facebook o Instagram. Esto implica la extracción de tendencias y patrones a partir de las publicaciones y comentarios hechos por los usuarios. En el ámbito político, el análisis de sentimientos en las redes sociales ha demostrado ser útil para evaluar la popularidad de los candidatos y comprender las preocupaciones y necesidades de los votantes. Esta información puede resultar crucial para diseñar con éxito campañas políticas.

5. **Medicina personalizada:** es un área revolucionaria en la atención médica cuyo objetivo es adaptar los tratamientos y terapias a las características individuales de cada paciente. Para ello se nutre de datos genéticos y clínicos para identificar patrones que ayuden a predecir la reacción del paciente ante un tratamiento específico.

1.3. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) es un campo de estudio centrado en el desarrollo de sistemas que puedan analizar y comprender el lenguaje humano a partir de un texto libre.

El objetivo del PLN es extraer el significado y el contexto de los textos. Usando PLN podemos hacer tareas como resumen automático de textos, generación de textos, traducción de idiomas, análisis de sentimientos, reconocimiento del habla y clasificación de artículos por temáticas. Para ello, utiliza conceptos lingüísticos como nombres, verbos, adjetivos, etc. El PLN se enfrenta a retos como las numerosas ambigüedades que contiene el lenguaje natural, tanto de palabras con doble significado como de estructuras gramaticales o frases hechas.

Para llevar a cabo estas tareas, el PLN hace uso de *representaciones de conocimiento*, como diccionarios de palabras con su significado, conjuntos de reglas gramaticales, etc. También

se suele hacer uso de ontologías, diccionarios de sinónimos, abreviaciones, etc.

El procesamiento del lenguaje natural es un paso clave en el proceso de extracción de conocimiento a partir de texto en el KDT. En este contexto, el PLN se utiliza para transformar el texto en una representación estructurada y procesable, permitiendo así aplicar técnicas de minería de texto. Algunas tareas típicas del PLN en el KDT incluyen la tokenización (división del texto en unidades más pequeñas como palabras o frases), el etiquetado de partes del discurso, la eliminación de stopwords, la detección de entidades nombradas, el análisis de sentimientos y la extracción de información relevante.

1.4. Formas intermedias

Debido a que el texto se trata de información no estructurada, dentro de la etapa de Prepocesamiento del proceso KDT, es crucial convertir el texto a una representación o forma intermedia que se pueda manejar de forma computacional. Esto va desde dividir el texto en unidades mínimas hasta identificar relaciones semánticas entre palabras. Al mismo tiempo se debe procurar no perder la integridad del texto inicial, es decir, no perder información sobre las relaciones entre términos. A continuación veremos algunas de las formas intermedias comúnmente empleadas [2].

1.4.1. Palabra

La palabra es el elemento mínimo de estudio sobre un texto. Por sí misma, una palabra no aporta suficiente información, pero algunas consideraciones como el conteo de palabras y la riqueza léxica (proporción de palabras diferentes con respecto al total) puede proporcionar información sobre el texto.

1.4.2. Bolsa de palabras

Consiste en recopilar todas las palabras que aparecen en un documento, ignorando el orden, la estructura gramatical y las relaciones semánticas. En esta representación, cada palabra se considera un *token* y se crea un vector que indica cuántas veces aparece cada palabra en el documento. Dentro de una bolsa podemos encontrar distintos tipos de tokens:

- **Palabra vacía** *(stop word)*: preposiciones, artículos y conjunciones, en definitiva son palabras que sirven para relacionar términos pero por sí mismas no aportan valor.
- **Palabra clave** *(keyword)*: se trata de palabras que identifican la temática del texto. Se les suele asociar un peso numérico para destacar su importancia.

A modo de ejemplo, podemos considerar el siguiente texto:

El perro persigue al gato. El gato huye del perro. (1.4.2)

La representación del texto mediante una bolsa de palabras podría ser:
La misma bolsa de palabras pero eliminado las *stop words*, se vería así:
Observamos que al representar el texto como una bolsa de palabras, se pierde completamente el orden. La bolsa de palabras nos dice cuántas veces aparece cada palabra, pero ya no sabemos nada sobre la relación entre el perro y el gato, y entre quién persigue y quién huye.

Palabra	Frecuencia
el	2
perro	2
persigue	1
al	1
gato	2
huye	1
del	1

Tabla 1.1.: Bolsa de palabras del ejemplo 1.4.2

Palabra	Frecuencia
perro	2
persigue	1
gato	2
huye	1

Tabla 1.2.: Bolsa de palabras sin *stop words* del ejemplo 1.4.2

Esta pérdida de contexto puede ser crítica en algunas tareas, ya que al perder la relación entre palabras, podemos perder información crucial.

1.4.3. Concepto

Un conepto es la representación mental de un objeto o una idea, expresada a través de un término. Los conceptos organizan palabras relacionadas bajo una idea común. Por ejemplo, el concepto de *animales domésticos* podría incluir palabras como perro, gato, pájaro, etc. Uno de los inconvenientes al tratar con conceptos es que a diferencia de las palabras, estos suelen dependen del contexto.

1.4.4. Taxonomía

Una taxonomía o jerarquía de conceptos se trata de una clasificación que organiza de forma ordenada conceptos o términos que se encuentran en niveles jerárquicos distintos, basándose en las relaciones entre ellos. Las taxonomías pueden representarse mediante jerarquías de términos, donde los conceptos más generales aparecen en los niveles superiores, y los conceptos más particulares aparecen en niveles inferiores. Un ejemplo de taxonomía del reino animal es el siguiente:

1. *Animalia*
    a) *Mammalia*
        I. *Felidae (Gatos)*
        II. *Canidae (Perros)*
    b) *Aves*
        I. *Accipitridae (Águilas)*

II. *Anatidae (Patos)*

c) *Reptilia*

I. *Pythonidae (Pitones)*

II. *Crocodylidae (Cocodrilos)*

1.4.5. Grafos

Un grafo o red semántica es una representación visual de conocimiento que emplea conceptos y las relaciones semánticas entre ellos. Estos conceptos son representados mediante nodos, y las relaciones mediante aristas o enlaces.

Una característica distintiva de las redes semánticas es el uso de la herencia, donde los nodos hijos heredan las características de los nodos padres. Estas redes son útiles para representar y comprender la semántica y las relaciones entre conceptos en un dominio específico. Además, destacan por ser capaces de preservar gran carga semántica, mientras que otras formas intermedias la pierden. Sin embargo, en ocasiones esto se traduce en un aumento excesivo de la complejidad del tratamiento de datos.

Entre los distintos tipos de red semántica destacan las redes IS-A y los grafos conceptuales.

1.4.6. Ontología

Las ontologías tiene una diversa cantidad de definiciones dentro de la literatura científica. En términos generales, se trata de mapa conceptual detallado y estructurado del conocimiento sobre un dominio específico. Una ontología está formada por los siguientes elementos básicos:

1. *Conceptos*: representan las entidades o clases dentro del dominio de interés. Por ejemplo, en una ontología sobre automóviles, podría haber conceptos como *Automóvil*, *Motor*, *Rueda*, etc.

2. *Roles*: describen relaciones binarias entre conceptos y propiedades. Por ejemplo, las propiedades *color*, *velocidad máxima*, *número de puertas* y las relaciones *es un tipo de*, *tiene*, etc.

3. *Individuos*: instancias o ejemplos.

4. *Axiomas*: especifican las reglas o restricciones que deben cumplirse en el dominio. Por ejemplo, *un automóvil tiene cuatro ruedas*.

Las ontologías proporcionan un marco formal para representar el conocimiento, lo que facilita la interoperabilidad entre distintos sistemas y permite a las máquinas realizar inferencias y razonamientos sobre el dominio de conocimiento especificado.

2. Spam

En este capítulo vamos a abarcar el concepto de Spam, los tipos de Spam, así como las distintas técnicas existentes en la literatura para detectarlo.

2.1. Definición e historia

El spam es el envío masivo y no solicitado de mensajes electrónicos, ya sea por correo electrónico, mensajes de texto, redes sociales y otros medios digitales. Generalmente los mensajes spam tienen como objetivo promocionar productos o servicios. A veces, estos mensajes también son usados para engañar a los usuarios para que revelen información personal, como contraseñas o información financiera. En contraste con el spam, el término *ham* se refiere a los mensajes de correo electrónico legítimos y deseados, que son enviados por remitentes conocidos o esperados. Estos mensajes suelen ser correos electrónicos personales, comerciales o informativos que se envían a destinatarios que han optado por recibirlos, por lo que no presentan ningún tipo de amenaza o molestia.

El término *spam* en el contexto de los mensajes de correo electrónico, se popularizó en la década de los años 90. El primer mensaje de spam conocido tiene su origen 1994, en un post de Usenet, donde una firma de abogados publicó un mensaje de anuncio de su firma legal. Desde entonces, la publicidad y el marketing mediante correo electrónico ha crecido a niveles impensables [4].

Según cifras del informe Kaspersky [5], se estima que en 2022 el 48.63 % de correos electrónicos en todo el mundo fueron spam.

2.2. Tipos de Spam

El Spam no es solo molesto para los usuarios, sino que también puede presentar una amenaza y un riesgo para la preservar la seguridad y la privacidad online. A continuación vamos a exponer los distintos tipos de Spam conocidos, según su naturaleza y el riesgo que representan [6]:

- **Correo electrónico no deseado:** es el tipo de spam más común y todos estamos habituados a encontrarlo. Con frecuencia inundan nuestra bandeja de entrada, resultando muy molesto al usuario. Tiene fines publicitarios o promocionales, por lo que no suelen representar un riesgo directo para la seguridad del usuario, aunque consumen recursos de almacenamiento.
- **Spam SEO:** también conocido como *spamdexing*, se refiere a la manipulación de los métodos de optimización de los motores de búsqueda para mejorar la clasificación de un sitio web y lograr que más usuarios accedan a él. Se puede clasificar en dos categorías:
  - **Spam de contenido:** los spammers llenan la página de palabras clave populares que el sitio web aparezca más arriba en las búsquedas.

    - **Spam de enlaces:** se trata de comentarios que podemos encontrar en foros o redes sociales, y que contiene un enlace externo que conduce a otro sitio web. Así se consigue atraer tráfico a la página web.

- **Spam en redes sociales:** se trata de perfiles y cuentas falsas creadas de forma masiva con el fin de propagar un mensaje spam, por ejemplo, propaganda de ideas de índole política o enlaces a sitios web externos.
- **Spam de mensajería:** es similar al spam por correo electrónico pero en SMS y en plataformas de mensajería instantánea tales como WhatsApp o Telegram.
- **Estafas de soporte técnico:** suelen comenzar con la llamada telefónica de alguien haciéndose pasar por empleado de una empresa. El estafador trata de convencer al usuario de que hay un problema con su cuenta o el producto contratado, pidiendole información sensible.
- **Spam de malware:** este spam contiene enlaces o archivos que pueden comprometer la seguridad del usuario simplemente con un click o una descarga. Suele llegar a través de un mensaje de texto o un correo electrónico no deseado.

2.3. Spam convencional y Phishing

A pesar del fastidio que representa el spam convencional, no suele resultar dañino pues su objetivo es anunciar productos o servicios. Sin embargo, el phishing es mucho más peligroso, pues supone el envío de correos fraudulentos que se hacen pasar por instituciones legítimas. Estos mensajes engañosos buscan obtener información confidencial, como contraseñas o números de tarjetas de crédito, con el fin de cometer robo de identidad o fraude financiero. Es de suma importancia que los usuarios estén capacitados para reconocer los patrones de estos intentos de phishing y que duden antes de enviar información sensible a terceros.

2.4. Técnicas de filtrado de Spam

Tal y como hemos mencionado en la anterior sección, detectar el spam es fundamental para evitar caer en fraudes y timos online. El spam, especialmente en forma de phishing, es hoy en día la herramienta principal usada por los estafadores para engañar a los usuarios y accede a su información confidencial.

Con el fin de proteger a los usuarios, los proveedores de servicios de electrónico han desarrollado durante estas décadas numerosas técnicas de detección de spam. En esta sección vamos a revisar el conjunto de técnicas de detección de spam que podemos encontrar en la literatura [7] [8]:

- **Listas negras y blancas:** también conocidas como *blacklists* y *whitelists*, este fue uno de los primeros métodos utilizados para detectar correo spam. Están basadas en la exclusión de mensajes que provienen de ciertos dominios, redes o servidores de Internet. Mediante este método se pueden identificar grandes cantidades de correo no deseado, aunque bien es cierto que es fácilmente falsificable y manipulable.
  Algunas de estas listas se encuentran en forma de ficheros de texto que contienen directamente remitentes o expresiones regulares de direcciones de correo electrónico.

Por el contrario, las listas blancas contienen direcciones de correo *verificadas* en las que se puede confiar. Este mecanismo, aunque simple, es muy difícil de burlar. Sin embargo, existe el inconveniente de que las empresas verificadas y exentas de estas listas pueden seguir enviando boletines de suscripción y publicidad en forma de spam.

- **Resúmenes:** son aplicaciones cuyo funcionamiento se basa en la creación de una representación compacta y única (resumen) de un correo electrónico. Se utilizan algoritmos de resumen, como *Nilsimsa*; estos algoritmos generan una firma única para cada mensaje basándose en su contenido, de manera que incluso pequeñas modificaciones en el mensaje producen cambio significativo en el resumen. Esto nos permite detectar variantes triviales de mensajes, como números aleatorios en el asunto o cambios mínimos en el contenido.

- **Modelos basados en contenido:** estas técnicas hacen uso del Machine Learning para determinar patrones y relaciones entre características de los mensajes Spam y Ham. Normalmente la forma de representación de los datos es mediante un vector de características por cada correo. Las características del vector contienen una lista de palabras representativas de la legitimidad de los mensajes.
  La elección de los términos más representativos de cada mensaje se realiza mediante técnicas de selección de características. La técnica más habitual es el cálculo de la ganancia de información (IG, *Information Gain*) de cada término con respecto a los posibles valores del atributo a predecir (spam o ham). Se acaba tomando los términos dentro del corpus con mayor ganancia de información.
  Otra de las técnicas comúnmente empleadas en el cálculo de representatividad de un término es la frecuencia de documentos que contienen un término dado (DF, *Document Frecuency*).

- **Técnicas basadas en análisis heurístico o en reglas:** A diferencia de los enfoques basados en aprendizaje automático, el análisis heurístico no requiere un conjunto de datos de entrenamiento, sino que se basa en el conocimiento previo de los patrones típicos de spam. Este enfoque usa reglas o heurísticas ya creadas para evaluar una gran cantidad de patrones, que suelen ser expresiones regulares. Si el email estudiado coincide con varios de los patrones evaluados, esto va aumentando su puntuación. A su vez, si alguno de los patrones no coincide, se resta puntuación. Se establece un umbral de Spam, y cualquier mensaje cuya puntuación lo supere se filtra como spam; de lo contrario se considera válido.
  Estas reglas están sujetas a actualizaciones frente a la amenaza de los spammers, que continuamente presentan nuevos tipos de mensajes spam que podrían pasar inadvertidos ante los filtros antiguos. Un ejemplo de filtro basado en reglas es *SpamAssassin* [9].

- **Métodos basados en casos:** el filtrado basado en ejemplos es una de las técnicas de filtrado de spam más populares. En este enfoque, se toman todos los correos de ambas clases (spam y ham) y se crea una colección. Luego se llevan a cabo los pasos de preprocesamiento para transformar el email, como selección de características, agrupación de datos, etc. Los datos son clasificados en dos conjuntos de vectores. Se utiliza un algoritmo de Aprendizaje Automático (basado en instancias) para entrenar datasets y testearlos para decidir si son Spam o Ham.

Podemos encontrar en la literatura existente numerosos métodos de filtrado de Spam en emails. De entre las técnicas anteriomente mencionadas, algunas de ellas aplican distintos

algoritmos de *Machine Learning*. A continuación vamos a exponer varias de las distintas técnicas de filtrado de spam que han sido usadas con éxito en los últimos años [7].

- **Naive Bayes**: es el algoritmo de Aprendizaje Automático más conocido en clasificación de textos. Este modelo ha adquirido gran popularidad por su capacidad de representar de forma simple y eficiente distribuciones complejas de probabilidad, además de su fácil implementación y rápida convergencia. El algoritmo se basa en el Teorema de Bayes y asume la independencia de los atributos, lo cual es una simplificación poco realista; no obstante, ha demostrado gran efectividad en numerosos estudios [10]. Se puede usar para resolver problemas de clasificación de dos o más clases, y maneja tanto datos continuos como discretos (siempre y cuando sean numéricos).

- **Máquinas de soporte vectorial** (SVM, *Support Vector Machines*): este tipo de modelos destaca por su sólida base teórica. El algoritmo se basa en la transformación de los datos existentes para encontrar un hiperplano de mayor dimensión donde maximinzar la separación existente entre las clases. Cabe destacar que con SVM no es necesario realizar selección de características en la fase de Preprocesamiento, pues su capacidad de aprendizaje no se ve afectada por haber demasiados atributos.

- **Boosting de Árboles de Decisión:** se basa en tomar varios algoritmos de aprendizaje débiles y combinar las hipótesis generadas en una única hipótesis de gran precisión. Para ello, el algoritmo se ejecuta varias veces sobre distintos subconjuntos de entrenamiento. Normalmente en filtrado de spam se combinan algoritmos como AdaBoost y C4.5.

- **Random Forest**: es un modelo basado en una colección de árboles de decisión, entrenado cada uno a partir de un subconjunto del corpus, de forma aleatoria. Para clasificar un correo nuevo, se pasan las características del correo a cada árbol, y estos emiten cada uno un voto unitario; entonces se le asigna al correo la clase con mayor número de votos. En este algoritmo es crucial tomar un número adecuado de atributos, pues afecta de forma directamente proporcional a la correlación y a la fortaleza. Para su ajuste se suele emplear la tasa de error. Los estudios de filtrado de spam realizados con este clasificador han demostrado obtener un alto grado de precisión [11].

- **k-NN (*K-Nearest Neighbors*)**: es uno de los algoritmos más empleados en clasificación basada en casos o instancias. En este método, cada ejemplo de correo electrónico se representa como un punto en un espacio multidimensional, donde las dimensiones son las características del correo electrónico (por ejemplo, palabras clave, frecuencia de palabras, etc.). Luego, cuando llega un nuevo correo electrónico, se compara con los ejemplos de entrenamiento y se clasifica según la mayoría de los k correos electrónicos más cercanos en términos de similitud. Si la mayoría de los k vecinos son correos electrónicos de spam, entonces el nuevo correo electrónico también se clasificará como spam. Este método es eficaz para detectar patrones sutiles en los datos y es relativamente simple de implementar.

Como hemos visto, el filtrado de emails *automático* parece ser actualmente el enfoque más exitoso para el filtrado de spam. Hace años, la mayor parte del correo no deseado podía detectarse simplemente analizando las direcciones de remitente y los asuntos de los emails, mediante listas negras. Sin embargo, los spammers adoptaron técnicas más sofisticadas como

el uso de direcciones arbitrarias y la inserción de caracteres al principio o final de la línea de asunto del mensaje.

Actualmente existen un gran número de filtros que hacen uso de una combinación de *Machine Learning* y conjuntos reglas generadas a partir de la Ingeniería de Conocimiento. El problema principal de utilizar reglas, es que este método no garantiza un resultado eficiente, ya que en primer lugar se debe generar un conjunto reglas (esto requerirá la ayuda de expertos en el tema) y en segundo lugar es preciso actualizar continuamente dicho conjunto. Esto puede llevar a una pérdida de tiempo y no es adecuado especialmente para usuarios inexpertos. En cambio, aplicando *Machine Learning*, no se requiere especificar ninguna regla, sino que se proporciona un conjunto de muestras de entrenamiento que son emails previamente clasificados. Por tanto el enfoque de Aprendizaje Automático ha demostrado ser más eficiente que el enfoque de Ingeniería de Conocimiento, al menos con respecto al esfuerzo que implica construir el sistema de filtrado.

3. Fundamentos matemáticos

Aquí se explican las bases matemáticas de los algoritmos de clasificación empleados
-conceptos básicos de probabilidad
-redes bayesianas
-algoritmos Naive Bayes
-SVM?

4. Experimentación

Tras haber estudiado las bases matemáticas de diversos algoritmos de aprendizaje supervisado, vamos a aplicar dichas técnicas a un caso real.

4.1. Dataset

Para la fase de experimentación he decidido tomar como corpus una base de datos de Kaggle [12]. En esta se combinan mensajes de correo electrónico de tres fuentes distintas:

- **`LingSpam.csv`**: este dataset es uno de los más antiguos conocidos a nuestra disposición y está constituido por 2591 mensajes de spam y ham recuperados de Linguist List [13]. Estos mensajes se centran en ofertas de trabajo, oportunidades de investigación y discusiones sobre software.
- **`EnronSpam.csv`**: el dataset está formado por 9687 correos de varios trabajadores de la empresa Enron Corporation. El dataset original contiene aproximadamente 500000 emails y se puede encontrar en [14]
- **`SpamAssassin.csv`**: este dataset recoge 6045 mensajes donados por varios usuarios de un foro público. Al tratarse de usuarios variados, estos mensajes son de temas menos específicos que aquellos mensajes que podrían provenir de un mismo usuario. El dataset original se puede encontrar en [15]

Los datasets que vamos a utilizar son simplificaciones de los originales, proporcionados por el dataset de Kaggle. Esta decisión de coger mensajes de distintas fuentes se ha tomado para fomentar la diversidad de contenido y que el modelo esté entrenado sobre una mayor variedad de escenarios. Por ejemplo, si sólo empleáramos la base de datos de EnronSpam, estaríamos limitando el espacio de detección del modelo al del ámbito empresarial.

Los tres archivos .csv constan de los siguientes atributos:

- **`id:`** identificador de cada instancia o mensaje.
- **`Body:`** cuerpo o contenido del mensaje.
- **`Label:`** clase a la que pertenece la instancia (toma el valor 1 si es spam y 0 si no es spam).

4.2. Definición del problema

Nos encontramos pues ante un problema de clasificación binaria, al cual vamos a aplicar varios algoritmos de aprendizaje supervisado para discernir qué técnica es capaz de identificar mejor las relaciones entre los contenidos de los mensajes.
Vamos a dividir el Dataset en un conjunto de entrenamiento para entrenar el modelo y un

conjunto de prueba para constatar la eficacia del modelo. El objetivo de esta fase va a ser entrenar varios modelos que puedan discernir con gran exactitud entre correo Spam y Ham, tanto para el conjunto de prueba como para un nuevo conjunto creado por nosotros y ajeno a las tres fuentes expuestas en 4.1.
La división del Dataset la vamos a hacer mediante la función `train_test_split` de Skicit-learn (ver Apéndice A.2). El conjunto de entrenamiento contendrá el 80 % de los datos y el conjunto de test el 20 % de los datos.

4.3. Preprocesado de los datos

Antes de entrenar el clasificador, es crucial realizar el preprocesado de los datos para garantizar la máxima eficacia del algoritmo de aprendizaje automático seleccionado. Se han empleado las siguientes técnicas de preprocesado:

- **Eliminación de valores nulos y duplicados:** Haciendo uso de los métodos `drop_duplicates()` y `dropna()` de la biblioteca `Pandas` (Apéndice A.1), podemos eliminar las filas con valores nulos (campos vacíos) y las filas duplicadas en un DataFrame.

- **Tokenización:** utilizamos la función `CountVectorizer()` de `sklearn` (Apéndice A.2) para convertir el cuerpo de cada correo electrónico en una representación numérica, donde cada palabra se convierte en una característica y se cuenta su frecuencia en cada correo electrónico.

  Realizamos este proceso porque el clasificador de Naive Bayes multinomial está pensado para aplicarlo en clasificación con características discretas, como recuentos de palabras [16]. Los datos de entrenamiento y de prueba se deben pasar al clasificador bien como una matriz densa (array-like) o una matriz dispersa (sparse matrix).

  La función de tokenización `CountVectorizer()` de `sklearn` precisamente devuelve una matriz dispersa, que ocupa menos espacio en memoria que una matriz densa o array-like. Hemos elegido esta función porque el conjunto de datos, al ser texto, es muy disperso, lo que significa que la mayoría de entradas en la matriz serán ceros.

- **Filtrado de palabras poco frecuentes:** eliminamos aquellas palabras que aparecen solo una vez en todo el corpus. Para ello vamos a usar el parámetro `min_df` en la configuración de `CountVectorizer()`. Este parámetro nos permite establecer un umbral mínimo del número de documentos en los que tiene que aparecer una palabra para poder formar parte del vocabulario. Al establecer `min_df=2`, estamos eliminando todas las palabras que aparezcan sólo una vez en todo el corpus. Con ello conseguimos reducir notablemente la matriz dispersa y quitar palabras irrelevantes, a la vez que mejoramos la eficiencia.

- **Eliminación de palabras vacías:** Las palabras vacías, tales como artículos, pronombres y preposiciones, se deben eliminar del texto antes de la tokenización. Esto se realiza con la intención de reducir el ruido en los datos y mejorar la calidad de las características tomadas. Para ello, scikit-learn proporciona una lista predeterminada de palabras vacías en inglés, denominada `ENGLISH_STOP_WORDS`, la cual podemos actualizar con palabras adicionales según sea necesario. Para especificar esta lista de palabras vacías en la configuración de `CountVectorizer()`, podemos utilizar el parámetro `stop_words=stop_words_list`.

Además, al analizar los correos electrónicos, se observó que la palabra *Subject* aparece al principio del cuerpo en todos los correos, lo cual es redundante y puede eliminarse.

4.4. Visualización y análisis exploratorio de los datos

-selección de características
-mapa de palabras
-balanceo de clases
-distribución de los datos

4.5. Algoritmos elegidos

-Naive Bayes Multinomial
-SVM

5. Interpretación de los resultados

...

5.1. Métricas de evaluación

Tomando como positiva la clase **Spam** y como negativa la clase **Ham**, vamos a definir varias de las métricas que nos servirán para evaluar el rendimiento y la eficacia del modelo construido.

En primer lugar representamos la matriz de confusión, que nos proporcionará una visión general del rendimiento del modelo, al mostrar cuántas instancias se han clasificado correctamente e incorrectamente en cada clase. La matriz se compone de las siguientes cuatro medidas:

- **TP (verdadero positivo):** el número de instancias *Spam* que han sido correctamente etiquetadas como *Spam*.
- **FP (falso positivo):** el número de instancias *Ham* que han sido incorrectamente etiquetadas como *Spam*.
- **TN (verdadero negativo):** el número de instancias *Ham* que han sido correctamente etiquetadas como *Ham*.
- **FN (falso negativo):** el número de instancias *Spam* que han sido incorrectamente etiquetadas como *Ham*.

		Clase real	
		Positiva	Negativa
Clase predicha	Positiva	Verdaderos Positivos (TP)	Falsos Positivos (FP)
	Negativa	Falsos Negativos (FN)	Verdaderos Negativos (TN)

Tabla 5.1.: Matriz de confusión

A partir de esta matriz, podemos calcular diversas métricas de evaluación del modelo:

- **Sensibilidad (*recall*):** También conocida como *Tasa de verdaderos positivos (TPR)*, es utilizada para conocer la proporción de instancias Spam que son etiquetadas correctamente.

$$Sensibilidad = \frac{TP}{TP + FN}$$

- **Especificidad (*specificity*):** También conocida como *Tasa de verdaderos negativos (TNR)*, es utilizada para conocer la proporción de instancias Ham que son etiquetadas correctamente.

$$Especificidad = \frac{TN}{TN + FP}$$

- **Precisión (*accuracy*):** Es la proporción de instancias (tanto Spam como Ham) que han sido etiquetadas correctamente.

$$Precisión = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Exactitud (*precision*):** Es la proporción de instancias Spam bien clasificadas entre todos los casos clasificados como positivos.

$$Exactitud = \frac{TP}{TP + FP}$$

- **Tasa de error (*error rate*):** Es la proporción de instancias (tanto Spam como Ham) que han sido etiquetadas incorrectamente.

$$ERR = \frac{FP + FN}{TP + TN + FP + FN}$$

- **F1-Score:** es una métrica muy utilizadas en problemas desbalanceados. Se construye a partir de la media armónica de la exactitud y la sensibilidad (recall).

$$F1\text{-}score = 2 \cdot \frac{exactitud \cdot recall}{exactitud + recall}$$

...

A. Bibliotecas usadas

A.1. Pandas

Pandas [17] es una biblioteca de Python utilizada principalmente para manipulación y análisis de datos. Ofrece estructuras de datos flexibles y herramientas para trabajar con datos tabulares y de series temporales. Estas funciones son fundamentales para realizar tareas comunes de limpieza y manipulación de datos. Algunas de las funciones que hemos empleado son:

- `read_csv`: Una función utilizada para leer datos desde archivos CSV y cargarlos en un DataFrame de Pandas. Esto permite la manipulación y análisis de datos de manera conveniente en Python.
- `drop_duplicates`: Una función utilizada para eliminar filas duplicadas de un DataFrame. Esto puede ser útil cuando se trabaja con conjuntos de datos que pueden contener duplicados y se desea mantener solo una instancia de cada fila única.
- `dropna`: Una función utilizada para eliminar filas o columnas que contienen valores nulos (NaN) en un DataFrame. Esto es útil para limpiar los datos y prepararlos para su análisis, eliminando observaciones incompletas o no válidas.

A.2. Scikit-learn

Scikit-learn [18] es una biblioteca de aprendizaje automático de Python que ofrece una amplia gama de herramientas para tareas como clasificación, regresión, clustering y más. A continuación, enumeramos algunas de las funciones que hemos empleado en este trabajo:

- `train_test_split`: Una función utilizada para dividir los datos en conjuntos de entrenamiento y prueba.
- `CountVectorizer`: Una función utilizada para convertir una colección de documentos de texto en una matriz de recuentos de términos/palabras.
- `MultinomialNB`: Una implementación del clasificador Naive Bayes multinomial en scikit-learn, adecuado para clasificación de textos con características discretas como recuentos de palabras.
- `ENGLISH_STOP_WORDS`: Una lista predeterminada de palabras vacías en inglés, como artículos, pronombres y preposiciones, utilizada para la eliminación de palabras vacías en el procesamiento de texto.

Glosario

La inclusión de un glosario es opcional.
Archivo: `glosario.tex`

$\mathbb{R}$ Conjunto de números reales.

$\mathbb{C}$ Conjunto de números complejos.

$\mathbb{Z}$ Conjunto de números enteros.

Bibliografía

[1] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, nov 1996.

[2] María del Consuelo Justicia de la Torre. Nuevas técnicas de minería de textos: Aplicaciones, 2017.

[3] María Novo-Lourés, Reyes Pavón, Rosalía Laza, David Ruano-Ordas, and Jose R Méndez. Using natural language preprocessing architecture (NLPA) for big data text sources. *Sci. Program.*, 2020:1–13, 2020.

[4] Wikipedia contributors. Correo basura. https://es.wikipedia.org/w/index.php?title=Correo_basura&oldid=157238043. Accessed: NA-NA-NA.

[5] Tatyana Kulikova. El spam y el phishing en 2022. https://securelist.lat/spam-phishing-scam-report-2022/97582/, February 2023. Accessed: 2024-3-4.

[6] Ivan Belcic. Qué es el spam: guía esencial para detectar y prevenir el spam. https://www.avast.com/es-es/c-spam, January 2020. Accessed: 2024-3-4.

[7] Florentino Díaz Fernando Corchado Juan M. Méndez, José R. Fdez Riverola. Sistemas inteligentes para la detección y filtrado decorreo spam: una revisión. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 2007.

[8] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, 5(6):e01802, 2019.

[9] Apache SpamAssassin: Welcome. https://spamassassin.apache.org/. Accessed: 2024-2-26.

[10] Ion Androutsopoulos, Georgios Paliouras, Eirinaios Michelakis, and Eirinaios Michelakis. Learning to filter unsolicited commercial e-mail. 2006.

[11] Gordon Rios and Hongyuan Zha. Exploring support vector machines and random forests for spam detection. 01 2004.

[12] User: nitishabharathi. Email spam dataset. https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset, 2020. Kaggle Dataset. Accessed: 2024-1-30.

[13] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, Georgios Paliouras, and Costantine Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *CoRR*, cs.CL/0006013, 01 2000.

[14] Leslie Kaelbling and Melinda Gervasio. Enron email dataset. https://www.cs.cmu.edu/~./enron/. Accessed: 2024-1-30.

[15] Spam assassin dataset. https://spamassassin.apache.org/old/publiccorpus/. Accessed: 2024-1-30.

[16] Sklearn.Naive_bayes.MultinomialNB. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Accessed: 2024-2-9.

[17] DataFrame — pandas 2.2.0 documentation. https://pandas.pydata.org/docs/reference/frame.html. Accessed: 2024-2-11.

[18] Scikit-learn. https://scikit-learn.org/stable/index.html. Accessed: 2024-2-9.