Universidad Nacional de Rosario

Tesina de Grado
para la obtención del grado de
Licenciado en Ciencias de la Computación

Sistemas de Apoyo a la Ingeniería Legal

Autor:
Luciano Francisco Perezzini

Directora:
Dra. Ana Casali
Co-Directora:
Dra. Claudia Deco

Departamento de Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Av. Pellegrini 250, Rosario, Santa Fe, Argentina

Resumen

Dentro de la ingeniería legal se necesitan de agentes artificiales con la capacidad de extraer conocimiento y patrones dentro de documentos legales para crear aplicaciones que asistan a profesionales a realizar ciertas tareas, muchas de las cuales necesitan ejecutarse en tiempo real, recuperando y analizando normativas de los boletines oficiales.

Actualmente, la búsqueda de normativas relevantes para determinadas actividades dentro de una empresa se realiza manualmente e involucra numerosos profesionales de distintas áreas dentro la misma. En el ámbito industrial, la anterior tarea es conocida como *matricería legal*. En este trabajo se propone un sistema de soporte en línea a la ingeniería legal con el objeto de transformar la matricería legal en una actividad semi-automática.

Dedicado a aquellas personas que confiaron en mí
cuando yo no pude

Agradecimientos

Quiero agradecer a las doctoras Ana y Claudia, quienes siempre supieron escucharme y brindar importancia a mis ideas y opiniones durante el desarrollo del presente trabajo. Gracias, también, por comprender mis inseguridades a lo largo de los últimos meses. Mis agradecimientos especiales a los doctores Guillermo Grinblat y Pablo Granitto quienes, aunque no estuvieron involucrados en la realización de esta tesina, supieron atender mis consultas con enorme paciencia, voluntad y respeto; además de comprender el esfuerzo que realizaba para, todos los días, aprender un poquito más con el propósito de lograr un trabajo eventualmente respetable.

Gracias a mis compañeros de la carrera por la confianza que últimamente me han brindado. Ellos saben quiénes son y no hace falta nombrarlos.

Finalmente, y lo más importante, gracias a mi familia: gracias por el coraje de acompañarme en cada particularidad a la que me enfrenté en estos veinte y tantos años. Gracias por permitirme terminar mis estudios de la mejor manera posible: en otras condiciones no lo hubiese podido lograr. Perdón por muchas veces estar ausente: es sólo mi forma de trabajar duro. Bien saben que no fue fácil.

Índice general

Capítulo 1

Introducción

> *The ultimate measure of a man is not where he stands in moments of comfort and convenience, but where he stands at times of challenge and controversy*
>
> — Martin Luther King, Jr.

En la era de la información, grandes volúmenes de datos se encuentran constantemente a nuestra disposición para analizarlos y eventualmente tomar decisiones (Elgendy et al. 2014). Hoy en día, casi cualquier aspecto de la sociedad moderna se ve impactado por los grandes datos: negocios, salud, administración y gobierno, entre otros (Wang et al. 2016). La habilidad de entender y crear valor dentro de largas cadenas de datos, generalmente no estructurados, provenientes de diversas fuentes de información, se ha convertido en una de las disciplinas más importantes dentro de cualquier entidad, principalmente dentro de empresas con fines de lucro (Merendino et al. 2018).

En la actualidad, las empresas exploran con gran detalle grandes volúmenes de información para intentar descubrir hechos que no conocían en el pasado, o intentar predecir futuros acontecimientos. La capacidad de aprovechar toda la información que se dispone es considerada como una habilidad crítica para el éxito organizacional en una empresa (Olszak 2016). Una empresa que dedica parte de sus ingresos a desarrollar esta capacidad, logra tomar decisiones más inteligentes y rápidas sobrepasando a sus competidores en el mercado.

Pero crear valor y descubrir conocimiento dentro de grandes datos no es una tarea sencilla. La variabilidad de las fuentes de información junto con la necesidad de combinar varias de estas y el uso de herramientas para encontrar patrones dentro de estos masivos conjuntos de datos se considera una tarea conjunta entre tecnólogos, científicos de datos y diversos departamentos dentro de una organización (Janssen et al. 2017). Como se explica en (García et al. 2016), se debe tener en cuenta que la calidad del conocimiento extraído depende en gran medida de la calidad de los datos de entrada: por lo general,

estos datos se ven afectados por factores negativos como el ruido, valores perdidos e inconsistencias.

Ahora bien, la información de texto juega un rol esencial en nuestras vidas, ya que nos comunicamos utilizando lenguajes naturales, producimos y consumimos enormes cantidades de datos textuales (páginas web, noticias, emails, etc.) todos los días y en diferentes circunstancias (Baggio 2016). La explosión de la información de texto, principalmente en la *World Wide Web* (o, simplemente, la *web*), hace que consumir y entender información en tiempo real sea una tarea prácticamente imposible para el ser humano. Con lo cual, se necesitan de agentes artificiales que asistan a la recuperación y análisis de información de forma rápida y precisa dentro de enormes repositorios de documentos de texto.

Particularmente, los documentos legales, como las normas legislativas y fallos judiciales, son documentos de texto escritos por el ser humano para la propia emisión en distintas comunidades dentro de un gobierno. Estos son de suma importancia para el orden de una comunidad de personas. Por naturaleza, la práctica de la Ley necesariamente involucra el análisis y la interpretación del lenguaje natural. Una actividad diaria la cual debe procesar largas cadenas de documentos legales y tomar decisiones acerca de la relevancia de los mismos acorde al contexto de una empresa es la llamada *matricería legal* y se presenta a continuación.

1.1. Matricería legal

Si vemos nuestra sociedad como un sistema, los documentos normativos forman parte de su especificación[1]. Una rama de las ingenierías que estudia estos sistemas es la denominada *ingeniería legal*. En (Shimazu et al. 2014) se define ingeniería legal como un campo de estudio que aplica ciencias de la información, ingeniería de software, e inteligencia artificial a documentos legales con el fin de apoyar a la legislación.

En especial, la matricería legal (ML) es una actividad que se considera perteneciente a la ingeniería legal y se realiza diariamente en distintas empresas con diversos contextos. Si bien no existe una definición formal y universal sobre esta actividad, se adopta, de ahora en más, la siguiente:

> *La ML es la actividad que se encarga de la compilación de normas legislativas exigibles a una entidad (organización o empresa) acorde con las actividades propias e inherentes de su actividad productiva.*

En pocas palabras, la ML en una empresa es el proceso de clasificar documentos normativos (leyes, decisiones administrativas, decretos, resoluciones,

[1]Se considera *especificación* de un sistema a descripciones abstractas del mismo

disposiciones, acordadas y todo acto administrativo publicado en boletines oficiales[2]) acorde a las tareas que ésta realiza. Aquí, el gran flujo de información consta de cientos de normativas publicadas de forma diaria por los distintos boletines oficiales del gobierno. Cabe destacar que la ML es una tarea verdaderamente importante: una errónea realización de la misma puede ocasionar desde penalizaciones del gobierno hasta la toma de malas decisiones en la realización de procesos industriales. Por lo general, esta actividad es realizada de forma periódica y cautelosa por un conjunto de profesionales (ingenieros ambientales, ingenieros industriales, abogados y documentalistas, entre otros) expertos en las actividades que la empresa realiza en la cotidianidad.

Como se detalló anteriormente, las fuentes de información que se utilizan a la hora de realizar la ML son los distintos boletines oficiales del gobierno; es decir, el Boletín Oficial de la República Argentina, los boletines oficiales provinciales, y los boletines oficiales municipales. Estas fuentes son las encargadas de publicar, electrónicamente, documentos normativos correspondientes a sus respectivas jurisdicciones.

Los profesionales encargados en realizar la ML en una empresa poseen un basto conocimiento acerca del contexto[3] de la misma y, por lo tanto, brindan importancia a ciertas ramas del Derecho[4] las cuales se consideran que están frecuentemente relacionadas con las actividades que la respectiva empresa ejecuta. Este punto es sumamente importante de destacar ya que cuando se posee un nuevo conjunto de normativas a catalogar, los profesionales encargados de la realización de la ML prestan atención, primeramente, a aquellas que pertenecen a determinadas ramas del Derecho sustanciales para la empresa. Luego, los mencionados expertos, corroboran que las normativas seleccionadas en el paso anterior sean verdaderamente relevantes para el contexto de la empresa; es decir, evalúan la importancia de las normativas con respecto a ciertas temáticas las cuales, por lo general, se encuentran estrechamente relacionadas con las actividades ejecutadas por ésta. A continuación, a modo de resumen, se enumeran los pasos realizados diariamente por los profesionales encargados de la ML en una empresa e:

1. Dependiendo la jurisdicción de e, se recuperan todos los documentos normativos del día de la fecha de distintos boletines oficiales.

2. Se seleccionan aquellas normativas, recuperadas del paso anterior, que se consideran pertenecientes a ciertas ramas del Derecho de principal interés para e.

[2]Comúnmente, se hará referencia a los documentos normativos como *normativas*

[3]Según la Real Academia Española, el *contexto* es "el entorno físico o de situación, político, histórico, cultural o de cualquier otra índole, en el que se considera un hecho"

[4]En esta tesina, se consideran ramas del Derecho a las divisiones del Derecho en distintas subdisciplinas las cuales atienden a distintos criterios, como su ámbito de aplicación

3. De los documentos anteriores y considerando específicamente las actividades que e ejecuta, los expertos realizan una evaluación de contenido más detallista con el objeto de aplicar una selección de normativas más refinada.

4. Las normativas seleccionadas del paso anterior son finalmente almacenadas en un repositorio el cual se denomina *matriz legal* o *matriz de requisitos legales* de e.

Mientras se construye y actualiza diariamente la matriz legal de una empresa, se debe realizar un seguimiento de las normativas pertenecientes a dicha matriz para verificar si, efectivamente, dicha empresa está cumpliendo con todos y cada uno de los documentos que se encuentran en la matriz.

La ML es una actividad engorrosa de realizar para el ser humano dado que, además de ser un ejercicio diario, cada boletín oficial publica alrededor de 60 normativas diarias. La clasificación de los anteriores documentos normativos en una empresa, actualmente, depende exclusivamente de un conjunto de profesionales especialistas en distintas áreas del Derecho, así como también en diversas actividades ejecutadas por ésta. El número de profesionales involucrados en la ML tiende a crecer en función de la magnitud de la empresa. Este proceso, por lo tanto, se considera costoso en tiempo y dinero.

Con la finalidad de dar soporte a los expertos encargados de desarrollar la ML en la actualidad, en esta tesina se propone un sistema de recomendación, en tiempo real, de documentos normativos utilizando técnicas de recuperación de información de texto y procesamiento del lenguaje natural. Este sistema tiene como objeto intentar emular el procedimiento que realiza todo experto encargado de la ML en una empresa. Se podrá notar, más adelante en el trabajo, que la cuestión más problemática a la hora de construir un sistema de estas características es cómo identificar y modelar el contexto de una empresa para luego utilizarlo artificialmente en la evaluación de relevancia de documentos normativos.

Durante el desarrollo de esta tesina, se destaca que no se han podido encontrar servicios informáticos comerciales, en la Argentina, que solventen el problema de la ML de forma automática o semi-automática. Se debe mencionar que se ha obtenido información sobre la existencia de empresas particulares que realizan la ML de forma manual mediante grupos de aproximadamente 10 personas por rama del Derecho.

Ahora bien, en los últimos años la ingeniería legal ha motivado científicos en diversos puntos del planeta y se han estudiado e implementado distintas propuestas. A continuación, se presentan algunas de ellas.

1.2. Estado del arte

Como se ha mencionado, se considera que el instrumento principal e indispensable de la Ley es el lenguaje natural. Mediante el lenguaje natural, los profesionales de la Ley comunican normativas las cuales debemos atender para pertenecer a una comunidad la cual mantenga la estabilidad social. Por lo tanto, no es sorprendente que la gran mayoría de las aplicaciones de la ingeniería legal tengan como núcleo al procesamiento del lenguaje natural (PLN), área que se introduce más adelante en el trabajo.

La aplicación del PLN, junto con otras técnicas de la inteligencia artificial en el ámbito legal no es algo nuevo. Los primeros sistemas para la búsqueda en línea de contenido legal aparecieron entre los años 1960s y 1970s; mientras que los agentes legales artificiales fueron un tópico de discusión fuerte entre los años 1970s y 1980s (Dale 2019).

La necesidad de intercalar el razonamiento legal con aquellos métodos computacionales como la minería de datos y el análisis de grandes colecciones de documentos legales ha provocado, en los últimos años, un gran interés entre investigadores de la inteligencia artificial y profesionales de la Ley. Esto ha generado la implementación de sofisticadas técnicas del PLN para ser aplicadas en grandes volúmenes de información jurídica y de esta forma, brindar a la sociedad herramientas simples para una mejor comprensión de la información legal disponible en diversas fuentes gubernamentales de libre acceso (Robaldo et al. 2019).

En la actualidad, es posible encontrar diversas aplicaciones del PLN a documentos legales: clasificación, representación de conocimiento, extracción de información, recuperación de información y población de ontologías. A continuación se nombran algunos trabajos relevantes:

- Según una reciente encuesta sobre el comportamiento de la lectura de políticas de servicio en contratos en la web, se reveló que los consumidores rara vez leen en profundidad contratos que requieren de su aceptación. En (Lippi et al. 2019) se propone un sistema de aprendizaje automatizado con la intención de detectar cláusulas injustas dentro de largos contratos electrónicos.

- Durante el proceso de analizar un caso particular, los profesionales de la Ley (abogados y jueces, entre otros) dependen de información acerca de casos similares. Estos documentos son largos y lingüísticamente complicados; por lo tanto, leer cada uno de ellos de forma detallada se considera una tarea particularmente desafiante. Con el objeto de afrontar este problema, en (Yamada et al. 2019) se propone un método de resumen de documentos legales.

- En (Neil et al. 2019) se presenta un enfoque para modelar argumentos legales, de potencial uso para la defensa o por la fiscalía durante un

juicio, utilizando redes Bayesianas.

- En (Waltl et al. 2019) se compara el rendimiento de la clasificación de normativas del Derecho Civil alemán utilizando distintas técnicas de clasificación de texto.

Los trabajos anteriores son sólo un breve resumen de la importancia y el enorme potencial de la ingeniería legal en la actualidad en diversos dominios tales como la sociedad en general y en industrias.

Con el propósito de reunir investigadores y profesionales de todo el mundo que desarrollan técnicas de PLN en el dominio legal, se organizó (por primera vez, en el mes de junio de 2019 en Estados Unidos) el taller *Natural Legal Language Processing*[5] en el cual participaron lingüistas computacionales, científicos de datos, profesionales de la inteligencia artificial y profesionales de la Ley, entre otros, para debatir nuevos problemas de la ingeniería legal y cómo afrontarlos utilizando, principalmente, herramientas del PLN.

[5] `http://nllpw.org`

Capítulo 2

Conceptos preliminares

A continuación, se introducen conceptos y definiciones de importancia para lo que resta del trabajo.

En la Sección 2.1 se introducen conceptos generales de los sistemas de información de texto, principalmente un modelo algebraico utilizado para representar documentos de forma vectorial. En la Sección 2.2 se realiza un breve resumen acerca de principales nociones del aprendizaje automatizado. Estos conceptos son necesarios de aplicar para poseer la potencial habilidad de entender grandes conjuntos de datos. Finalmente, en la Sección 2.3, se presentan los sistemas de recomendación. Estos sistemas utilizan técnicas de las anteriores secciones con el objeto de sugerir información potencialmente relevante para un usuario en particular.

2.1. Sistemas de información de texto

Antes de introducir los sistemas de información de texto (SIT), es conveniente presentar conceptos generales de la recuperación de información (RI).

La RI es un área de estudio relevante en la era de la información y toma muchas y diversas formas en la cotidianidad: se considera un área de estudio dentro de las ciencias de la computación y se encarga de la representación, almacenamiento, organización y el acceso a ítems de información (Baeza-Yates et al. 1999). Un *ítem* puede tomar cualquier forma: documentos de texto, archivos de audio, imágenes, entre otras. La RI trata, por lo general, con información no estructurada: información que no posee esquemas bien definidos y consecuentemente, no se considera relativamente fácil de manipular por una máquina.

El objetivo de un sistema de RI es asistir al ser humano en encontrar información relevante dentro de grandes colecciones de datos, en tanto satisfaga su necesidad de información (Manning et al. 2008). Por lo general, en un sistema de estas características, los ítems son retornados por órden de relevancia; es decir, generalmente no existe un ítem que sea la respuesta exacta a cierta

consulta, a comparación de la tradicional recuperación de datos (Baeza-Yates et al. 1999). A modo de ejemplo, Google Search[1] es el sistema de RI más popular de la actualidad. Se destaca que los usuarios que hacen uso de un sistema de RI, por lo general, no suelen tener una idea bien formada acerca de la información que necesitan encontrar. Por lo tanto, estos últimos no tienen la capacidad de expresar su particular necesidad de información en una estrategia de búsqueda correcta. Esta particularidad es uno de los mayores desafíos en el área de recuperación de información. Para afrontarlo existen diversas técnicas de aprendizaje, implícitas y explícitas, con el objetivo de refinar una consulta en base a la necesidad de información de un usuario. Entre las heurísticas más utilizadas para afrontar este problema se puede nombrar la denominada Retroalimentación de Relevancia (Ruthven et al. 2003). Resumidamente, es posible formalizar un sistema S de RI como sigue:

$$(I, q) \underset{S}{\longrightarrow} I_R$$

donde I es el conjunto de toda la información que se dispone, q es la necesidad de información del usuario resumida en una consulta, e $I_R \subset I$ es el conjunto de información que S considera como relevante ante q.

Ahora bien, un tipo de datos que juega un importante papel en grandes y conocidos repositorios de información como lo es la web, es la información textual. El texto es la forma preferida de los seres humanos para expresar información (Baggio 2016). Por lo tanto, se necesita de agentes artificiales que ayuden a procesar y explotar enormes volúmenes de datos textuales. Primeramente, se debe poseer herramientas que puedan recuperar información de grandes repositorios de datos textuales para luego, emplear agentes artificiales que tengan la capacidad de descubrir conocimiento en información de texto y hacer uso del mismo en diversas aplicaciones. El comentario anterior se puede resumir mediante el siguiente diagrama:

$$I \underset{S}{\longrightarrow} I_R \underset{A}{\longrightarrow} \textit{conocimiento} \longrightarrow \textit{aplicaciones} \tag{2.1}$$

donde I es un repositorio (conjunto) de documentos textuales de gran volumen, S es un sistema de recuperación de texto, $I_R \subset I$ es una primera identificación de documentos recuperados de I potencialmente relevantes para una tarea en particular y A es un agente artificial con la capacidad de realizar una evaluación de texto más exhaustiva con el propósito de descubrir conocimiento y patrones. Con los conocimientos que se aprenden gracias a A, se construyen aplicaciones que asisten al humano para una mejor comprensión del contenido textual de los documentos en I_R. Por lo general, estas aplicaciones suelen estar enfocadas en la asistencia a usuarios con respecto a la toma de decisión.

[1] `http://google.com`

A diferencia de datos estructurados, que son aquellos que poseen esquemas bien definidos, el texto se considera desafiante a la hora de buscar patrones dentro del mismo. Como consecuencia, el análisis de texto automático se ha convertido en un área de estudio emergente dentro de la lingüística y las ciencias de la computación: el procesamiento del lenguaje natural (PLN) es una importante rama de estudio en la actualidad (Singh 2018) y se encarga de utilizar métodos computacionales para procesar el habla o el texto escrito en forma libre (Assal et al. 2011).

Ahora bien, un SIT es aquel que cumple con los dos primeros pasos del diagrama 2.1; es decir, posee las siguientes dos capacidades:

1. **Acceso a la información**. Esta primer etapa consiste en identificar, dentro de grandes colecciones de documentos de texto, documentos que se consideren potencialmente relevantes para la realización de cierta tarea de usuario. Aquí, simplemente, los documentos son retornados en su forma original: es decir, no se realiza ningún procedimiento de análisis de texto con el objeto de descubrir conocimiento para seguir asistiendo al usuario a digerir la información retornada. En estas circunstancias, con la falta de técnicas de análisis de texto, el usuario debe proceder con la ardua tarea de leer todos los documentos devueltos con la finalidad de descubrir información más específica (es decir, información más relevante con respecto su necesidad de información). Según (Cybenko et al. 1999), se puede clasificar al acceso de información en dos modos:

    - Pull: el usuario es quien inicia la búsqueda de información en el sistema mediante una consulta que aparenta representar su necesidad de información. El sistema espera a que el usuario realice una consulta para devolver documentos potencialmente relevantes. Se dice que el sistema juega un rol pasivo y que el usuario tiene una necesidad de información temporaria.
    - Push: aquí, es el sistema (agente inteligente) quien recomienda un conjunto de documentos al usuario. En este caso, la necesidad de información del usuario puede ser relativamente estable. Dado una cadena de documentos dinámica (es decir, aquella que cambia frecuentemente), el sistema es quien selecciona algunos de ellos en base a la necesidad de información estable del usuario y se los presenta como resultado final. Este modo de acceso es importante ya que se empleará para construir, más adelante, el sistema propuesto para afrontar la problemática de la matricería legal en la actualidad.

2. **Análisis de texto**. El análisis de texto es la capacidad de descubrir patrones de forma automática en documentos de texto, permitiendo al usuario descubrir conocimiento y tomar decisiones más rápidamente, sin la necesidad de leer cada uno de los documentos recuperados en el

ítem anterior. La clasificación de texto es una de las herramientas más importantes dentro del análisis de texto y se presenta una introducción en el Capítulo 3.

En lo que resta de la presente sección se hace foco en conceptos importantes acerca del acceso a la información de un SIT. Particularmente, se presenta el modelo de recuperación de texto denominado *Espacio-Vectorial*.

2.1.1. El modelo Espacio-Vectorial

La herramienta más importante en el acceso a la información de texto es el motor de búsqueda: éste brinda soporte para consultar y recuperar información en repositorios de documentos de texto y además puede ser fácilmente extendido para implementar un sistema recomendador. Generalmente, desde el punto de vista del usuario, el problema de la recuperación texto es el de utilizar una consulta para encontrar documentos relevantes en una colección.

Durante décadas, se han estudiado diversos modelos de recuperación de texto. Estos se pueden categorizar en tres ramas (Hai Dong et al. 2008): los basados en la teoría de conjuntos, los algebraicos, y los probabilísticos. Uno de los modelos de recuperación de texto más utilizado por su sencillez y efectividad es el modelo algebraico llamado Espacio-Vectorial (MEV) (Nath et al. 2013) (Aggarwal et al. 2012). El MEV se propuso por primera vez en (Salton et al. 1975), y se describe en la presente sección.

Primeramente, se definen los siguientes dos conceptos básicos de un sistema de información de texto en el cual, por simplicidad, se considera que un *término* es análogo a una palabra.

Definición 2.1.1 (Colección de documentos)**.** Se define por *colección de documentos*, o *corpus*, a un conjunto de documentos $\mathscr{C} = \{d_1, d_2, \ldots, d_n\}$, para algún $n \in \mathbb{N}$.

Definición 2.1.2 (Vocabulario)**.** Sea $\mathscr{C}$ una colección de documentos. Se define *vocabulario* de $\mathscr{C}$ ($V_{\mathscr{C}}$) al diccionario de todos los términos utilizados en la colección. Se nota t_i y $|V_{\mathscr{C}}|$ al i-ésimo término del diccionario y al número total de términos del mismo, respectivamente.

Dada una colección de documentos $\mathscr{C}$ y un documento d cualquiera, el MEV representa a d como un vector $\vec{v}_d$ donde cada dimensión corresponde a un término del vocabulario $V_{\mathscr{C}}$; por lo tanto, $\vec{v}_d$ es un vector $|V_{\mathscr{C}}|$-dimensional (Manning et al. 2008). A cada término t en d se le otorga cierto peso $w_{t,d}$ el cual, principalmente, depende del número de ocurrencias de t en d:

$$\vec{v}_d = (w_{t_1,d}, w_{t_2,d}, \ldots, w_{t_{|V_{\mathscr{C}}|},d}).$$

Se destaca que en el caso de que existan términos en d que no se encuentren en el vocabulario $V_{\mathscr{C}}$, éstos se descartan. Por lo general, $\vec{v}_d$ es ralo: los documentos

no utilizan todo el vocabulario de $\mathscr{C}$. Esta forma de representar documentos se denomina *bolsa de palabras*: aquí, el orden exacto de los términos en un documento es ignorado y sólo se retiene información acerca de la ocurrencia de los términos. En particular, si el procedimiento anterior se realiza para todo documento $d \in \mathscr{C}$, se obtiene una matriz término-documento, $M_{\mathscr{C}}$, la cual posee $|V_{\mathscr{C}}|$ filas y $|\mathscr{C}|$ columnas:

$$M_{\mathscr{C}} = \begin{pmatrix} w_{t_1,d_1} & w_{t_1,d_2} \cdots & w_{t_1,d_{|\mathscr{C}|}} \\ w_{t_2,d_1} & w_{t_2,d_2} \cdots & w_{t_2,d_{|\mathscr{C}|}} \\ \vdots & \vdots & \vdots \\ w_{t_{|V_{\mathscr{C}}|},d_1} & w_{t_{|V_{\mathscr{C}}|},d_2} \cdots & w_{t_{|V_{\mathscr{C}}|},d_{|\mathscr{C}|}} \end{pmatrix}$$

Una de las formas más sencillas de otorgar los anteriores pesos es mediante el cómputo del número de ocurrencias del término en el documento. Este esquema de pesaje se denomina *frecuencia de términos* y se nota $tf_{t,d}$. De esta forma:

$$\vec{v}_d = (tf_{t_1,d}, tf_{t_2,d}, \ldots, tf_{t_{|V_{\mathscr{C}}|},d}) \in \mathbb{N}^{|V_{\mathscr{C}}|}.$$

El esquema de pesaje anterior sufre del siguiente problema: todos los términos son considerados igualmente importantes cuando en realidad, sólo un puñado de éstos brindan información significativa sobre el contenido del mismo. Por lo tanto, a lo largo de las últimas décadas, se han propuesto diversas formas de refinar esquemas de pesaje para una mejor representación de documentos de texto en el MEV. El esquema más aceptado en la comunidad científica es calcular el peso de un término t mediante el producto de un peso local y un peso global (Manning et al. 2008):

$$w_{t,d} = w_{t,d}^{local} \times w_t^{global} \tag{2.2}$$

Una instancia del esquema de pesaje 2.2 es el denominado *frecuencia de término–frecuencia inversa de documento*[2] (*tf-idf*). Antes de presentar el anterior esquema, se definen los conceptos de frecuencia de documento y frecuencia de documento inversa.

Definición 2.1.3 (Frecuencia de documento)**.** Sean $\mathscr{C}$ una colección de documentos y t un término. La *frecuencia de documento* de t en $\mathscr{C}$ (df_t) se define como el número de documentos en $\mathscr{C}$ que contienen el término t. Es decir:

$$df_t = |\{d \in \mathscr{C} \mid t \in d\}|$$

Definición 2.1.4 (Frecuencia de documento inversa)**.** Sean $\mathscr{C}$ una colección de documentos y t un término. La *frecuencia de documento inversa* de t en $\mathscr{C}$ (idf_t) se define como sigue:

$$idf_t = \log\left(\frac{|\mathscr{C}|}{df_t}\right)$$

[2]Del inglés, *term frequency–inverse document frequency*

Luego, el esquema tf-idf se define como sigue:

$$\begin{aligned} tf\text{-}idf_{t,d} &= w_{t,d}^{local} \times w_t^{global} \\ &= tf_{t,d} \times idf_t \end{aligned} \tag{2.3}$$

Como se puede observar, la finalidad de *idf* en la definición 2.3 es reducir los pesos de los términos que aparecen en muchos documentos de $\mathscr{C}$, ya que se consideran menos informativos que los que aparecen en una porción más pequeña de la colección. Como se explica en (Manning et al. 2008), sean t un término y d un documento, entonces:

- $\uparrow\uparrow$ $tf\text{-}idf_{t,d}$ cuando t ocurre muchas veces dentro de un subconjunto pequeño de documentos de $\mathscr{C}$;
- $\downarrow$ $tf\text{-}idf_{t,d}$ cuando t ocurre pocas veces en d, o ocurre en muchos documentos de $\mathscr{C}$;
- $\downarrow\downarrow$ $tf\text{-}idf_{t,d}$ cuando t ocurre en, prácticamente, cada documento de $\mathscr{C}$.

Los términos "raros" son aquellos que poseen un peso superior al resto, ya que se consideran más relevantes y proporcionan más información acerca del contenido semántico del mismo. Por lo tanto, dado un documento d, se dice que el vector

$$\vec{v}_d = (tf\text{-}idf_{t_1,d}, tf\text{-}idf_{t_2,d}, \ldots, tf\text{-}idf_{t_{|V_{\mathscr{C}}|},d})$$

intenta capturar la relevancia de los términos de $V_{\mathscr{C}}$ que ocurren en el documento d. A lo largo de este trabajo se utiliza tf-idf como esquema de pesaje.

Ahora, dado que todo documento es representado mediante un vector $|V_{\mathscr{C}}|$-dimensional, la similitud del coseno se usa de manera estándar para cuantificar la similitud entre las representaciones vectoriales (Aggarwal et al. 2012) y es la cual se utiliza a lo largo de esta tesina. Sean d_1 y d_2 documentos cualesquiera, entonces:

$$sim(d_1, d_2) = sim(d_2, d_1) = cos(\vec{v}_{d_1}, \vec{v}_{d_2}) = \frac{\vec{v}_{d_1} \cdot \vec{v}_{d_2}}{\|\vec{v}_{d_1}\|\|\vec{v}_{d_2}\|} = \hat{v}_{d_1} \cdot \hat{v}_{d_2},$$

donde $\|\cdot\|$ es la norma euclídea. En la Figura 2.1 se ilustra una representación de la similitud entre dos documentos con un vocabulario de dos términos. La similitud del coseno resulta de gran interés en un motor de recuperación de texto por las siguientes razones:

- Rango acotado de similitud. Dado que toda componente de un vector representación de un documento es no negativo (por consecuencia trivial del esquema tf-idf), la similitud del coseno entre dos documentos cualesquiera pertenecerá al subconjunto de los números reales $[0, 1]$.

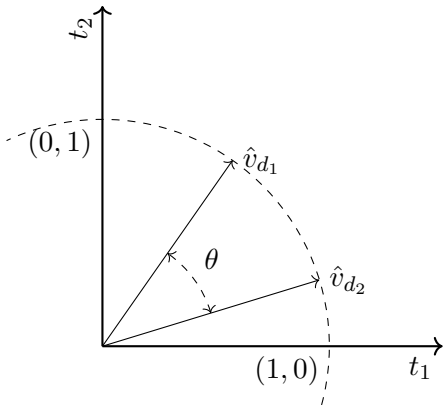

Figura 2.1: Ejemplo de representación, en el MEV, de dos documentos d_1 y d_2 con un vocabulario de dos términos: t_1 y t_2. Particularmente, los vectores representación de ambos documentos se encuentran normalizados con el propósito de compensar sus longitudes. $sim(d_1, d_2) = cos(\theta)$ representa la similitud semántica entre ambos documentos.

- Normalización de longitud de documentos. Sean d_1 y d_2 documentos, junto con sus respectivos vectores representación $\vec{v}_{d_1}$ y $\vec{v}_{d_2}$. Una forma trivial de cuantificar la similitud entre ambos documentos es calcular la magnitud del vector diferencia entre los vectores $\vec{v}_{d_1}$ y $\vec{v}_{d_2}$. Esto sufre de una gran desventaja: si se supone que d_1 y d_2 poseen un contenido similar pero d_1 es mucho más largo que d_2; entonces, la magnitud del vector diferencia entre ambos vectores será significativa, cuando en realidad ambos documentos son similares. Por definición, la similitud del coseno compensa el efecto de la longitud de los documentos dividiendo a cada uno de estos por su longitud.

Para finalizar, el MEV es un modelo que (1) *aprende* un vector $|V_{\mathscr{C}}|$-dimensional de pesos globales de términos ($\vec{\delta}_{\mathscr{C}}$) con el cual (2) *transforma* documentos a su representación vectorial. Sean $\mathscr{C}$ una colección de documentos y d un documento cualquiera, entonces:

$$\mathscr{C} \underset{aprender}{\longrightarrow} \vec{\delta}_{\mathscr{C}} = (w_{t_1}^{global}, \ldots, w_{t_n}^{global}) \ \forall \ t_i \in V_{\mathscr{C}}$$

$$d \underset{transformar_{\vec{\delta}_{\mathscr{C}}}}{\longrightarrow} \vec{v}_d = (w_{t_1,d}^{local}, \ldots, w_{t_n,d}^{local}) \odot \vec{\delta}_{\mathscr{C}} \ \forall \ t_i \in V_{\mathscr{C}}$$

donde $\odot$ es el producto de Hadamard[3] y $V_{\mathscr{C}}$ es el vocabulario de $\mathscr{C}$. Esta forma de tratar al MEV como un modelo de aprendizaje y transformación será de utilidad más adelante.

[3]Sean A y B dos matrices de dimensión $m \times n$, el producto de Hadamard $A \odot B$ es una matriz de misma dimensionalidad que A y B donde sus elementos se calculan de la siguiente forma $(A \odot B)_{i,j} = (A)_{i,j}(B)_{i,j}$

2.1.2. Preprocesamiento de texto

El objetivo principal del preprocesamiento de texto es determinar el vocabulario de términos (Manning et al. 2008). La mencionada tarea es realizada mediante una composición de operaciones lingüísticas y debe realizarse antes de recuperar y analizar documentos. Por lo general, el orden en el cual se aplican estas operaciones es el siguiente: (1) tokenización[4], (2) normalización, (3) supresión de palabras vacías y (4) stemming. Si d es un documento, el resultado de aplicar en orden los procedimientos anteriores es una lista de términos $[t_1, \ldots, t_n]$, para algún $n \in \mathbb{N}$. Estas operaciones se describen a continuación.

Tokenización

Es el procedimiento de cortar largas cadenas de texto para obtener pequeñas cadenas las cuales se denominan términos. El problema principal es determinar cuáles son los términos correctos a utilizar. Particularmente, en el idioma español, el procedimiento es sencillo: se descartan los espacios en blanco y los signos de puntuación. Esto no sucede en muchos otros idiomas (Kaplen 2005). Lo mismo sucede con el resto de las técnicas de preprocesamiento de texto. Notamos *tok* a la función tokenización.

Ejemplo.

$d =$ "Lo que uno puede llegar a ser, uno debe serlo."
$tok(d) =$ ['Lo', 'que', 'uno', 'puede', 'llegar', 'a', 'ser', 'uno', 'debe', 'serlo']

Normalización

La normalización toma cada término y lo lleva a una forma normal o canónica.

Ejemplo. Sea $t = U.N.R.$ un término que referencia a la Universidad Nacional de Rosario, luego:

$t \mapsto unr$

Dentro de la normalización, por lo general, se realizan las siguientes operaciones en conjunto para lograr un resultado final: transformación de todo token a minúsculas, supresión de valores numéricos y supresión de signos diacríticos[5]. Notamos *norm* a la función normalización de términos.

Ejemplo.

[4]Del inglés, *tokenization*

[5]El término *diacrítico* se refiere a un signo ortográfico que sirve para dar a una letra o a una palabra algún valor distintivo

$d =$ "Una buena decisión se basa en el conocimiento, y no en los números."
$(norm \circ tok)(d) =$ ['una', 'buena', 'decision', 'se', 'basa', 'en', 'el', 'conocimiento', 'y', 'no', 'en', 'los', 'numeros']

Supresión de palabras vacías

Las *palabras vacías*[6] son aquellos términos, de cierto lenguaje, que son extremadamente frecuentes y por lo tanto, la presencia de ellas en un documento es insignificante. Es decir, las palabras vacías de un lenguaje no brindan información acerca del contenido semántico de un documento (Rajaraman et al. 2011). Las siguientes son simplemente algunas de las palabras vacías del idioma español:

`de`, `la`, `que`, `el`, `en`, `y`, `a`, `los`, `del`, `se`, `las`, `por`, `un`, `para`, `con`, `no`, `una`

Notamos *rem* a la supresión de palabras vacías.

Stemming

El stemming es una técnica lingüística la cual se encarga de la estandarización y la correcta representación de los rasgos semánticos descriptores de los documentos (Eraso et al. 2011). En otras palabras, como se explica en (Manning et al. 2008), el stemming es el proceso que reduce un conjunto de palabras a su raíz léxica común.

Ejemplo.

`univers` es la raíz léxica de universidad, universidades y universo, entre otros.

Uno de los algoritmos de stemming más utilizado y el cual es aplicado a lo largo de este trabajo, es el *algoritmo de Porter* propuesto en (Porter 1997). Existen diversas modificiones del mencionado algoritmo para distintos idiomas, incluído el español[7]. Notamos *stem* a la función de stemming.

Ahora bien, se puede considerar al preprocesamiento de texto como la composición $preproc = stem \circ norm \circ rem \circ tok$. Es decir, dado un documento d,

$$\begin{aligned} preproc(d) &= (stem \circ norm \circ rem \circ tok)(d) \\ &= [t_1, \ldots, t_n], \text{ para algún } n \in \mathbb{N}. \end{aligned} \tag{2.4}$$

A modo de ejemplo, el siguiente es un caso de aplicación del preprocesamiento de texto a un documento concreto:

[6] Del inglés, *stop words*
[7] Proyecto *Snowball*: `https://snowballstem.org/`

d = "A veces sentimos que lo que estamos haciendo es sólo una gota en el océano. Pero si esa gota no estuviera en el océano, el océano sería menos por no tenerla."
$preproc(d)$ = ['sent', 'got', 'ocean', 'got', 'ocean', 'ocean', 'ten']

2.1.3. Propiedades del texto

A continuación, se mencionan dos importantes propiedades del texto cuando se emplea el MEV como modelo de recuperación:

- **Espacio vectorial de gran dimensionalidad**. Como se detalló anteriormente, todo vector representación de un documento es $|V|$-dimensional, donde cada dimensión representa un término del vocabulario. Para comprender la magnitud de $|V|$, existe una ley empírica denominada *ley de Heaps*. Esta ley estima el número de términos distintos en una colección de documentos:
$$|V| = kT^b,$$
donde T es el número de términos en la colección. Típicamente, $30 \leq k \leq 100$ y $b \approx 0.5$ (Manning et al. 2008).

- **Pocos términos irrelevantes**. Una forma de evitar espacios vectoriales de gran dimensionalidad es asumir que muchas de las características son irrelevantes y utilizar técnicas de selección de características para intentar excluir aquellas irrelevantes. Como se explica en (Joachims 2002), en el dominio del texto muy pocos términos se consideran irrelevantes y aplicar técnicas de selección de características puede provocar pérdida de información. Como se describe en (Aggarwal et al. 2012), en el texto, la remoción de palabras vacías y el empleo de *idf* se consideran técnicas de selección de características. De igual forma, las anteriores técnicas, no logran disminuir en gran porcentaje el número de términos del vocabulario y, por lo tanto, el espacio vectorial continúa siendo de gran dimensionalidad. Como se comentará en el Capítulo 3, esta particularidad de la representación de texto en el MEV, logra que sea atractiva para ciertos algoritmos de aprendizaje automatizado.

2.2. Aprendizaje automatizado

En la presente sección, se repasan nociones elementales del *aprendizaje automatizado*.

El aprendizaje automatizado es un área inherentemente multidisciplinaria que incluye nociones y resultados de áreas de estudio como la inteligencia artificial, probabilidad y estadística, teoría de la información y ciencias cognitivas, entre otras. Una de las definiciones más aceptadas en la literatura es la que se propone en (Mitchell 1997):

> "Se dice que un programa de computadora aprende de la experiencia E con respecto a un conjunto de tareas T y medida de rendimiento P, si su rendimiento en las tareas de T medidas por P, mejora con la experiencia E".

El aprendizaje automatizado es necesario en casos en los que no es directamente posible escribir un programa de computadora para resolver cierto problema y por lo tanto, la alternativa es aprender de ejemplos o experiencias pasadas (Alpaydin 2010).

En otras palabras, el aprendizaje automatizado consiste en programar computadoras para optimizar un criterio de rendimiento utilizando datos de ejemplo o experiencias pasadas: por lo general, se posee un modelo matemático definido hasta ciertos parámetros y el aprendizaje se define como la ejecución de un programa o procedimiento de computadora el cual optimiza los parámetros del modelo utilizando datos de entrenamiento o experiencia pasada (Alpaydin 2010). Al procedimiento anterior se lo denomina algoritmo de aprendizaje o método de aprendizaje.

2.2.1. Aprendizaje supervisado

En la práctica, por lo general, se dispone de un conjunto de n pares entrada-salida $\mathscr{D} = \{(\vec{x}_i, y_i)\}_{i=1}^{n}$ donde cada y_j fue generado por una función desconocida

$$y = f(x). \tag{2.5}$$

Se define observación a todo par $(\vec{x}_j, y_j) \in \mathscr{D}$, para algún $j \in \{1, \ldots, n\}$. Aquí, $\vec{x}_j$ se denomina vector de características. Un vector de características es un vector p-dimensional, para algún $p \in \mathbb{N}$, en el cual cada dimensión contiene un valor que describe el ejemplo de cierta forma. y_j se denomina etiqueta y es la medida resultado con respecto a $\vec{x}_j$. Esta medida puede ser categórica o cuantitativa. Esta distinción en el tipo de salida ha llevado a una convención de nomenclatura para tareas de predicción: se denominan problemas de *regresión* a aquellos que involucran respuestas cuantitativas y problemas de *clasificación* a aquellos que involucran respuestas cualitativas (Alpaydin 2010). El objetivo del aprendizaje supervisado es aplicar un algoritmo de aprendizaje al conjunto de entrenamiento para aprender una función, o modelo, $\hat{f}$ tal que sea una aproximación de la función 2.5 (Hastie et al. 2009). Para medir el desempeño de $\hat{f}$, ésta misma se aplica a un conjunto de prueba disjunto del conjunto de entrenamiento. La habilidad de predecir correctamente ejemplos del conjunto de prueba es conocida como generalización (Bishop 2006).

El aprendizaje se dice que es *supervisado* porque todo ejemplo $\vec{x}_j \in \mathscr{D}$ posee una medida de respuesta asociada y_j, con $j \in \{1, \ldots, n\}$. Mientras que en el denominado aprendizaje *no supervisado* los vectores características carecen de una respuesta asociada. Según (James et al. 2013), la mayoría de

los métodos de aprendizaje pueden ser categorizados como *paramétricos* o *no paramétricos*:

1. Los métodos paramétricos se basan en los siguientes dos procedimientos.

   a) Se asume la forma de f. Por ejemplo, si se asume la existencia de una relación lineal en el conjunto de entrenamiento, entonces f tendrá la forma:
   $$f(\vec{x}) = \vec{w}^T\vec{x} + b,$$
   donde $\vec{w} = (w_1, \ldots, w_p)^T$ y $\vec{x} = (x_1, \ldots, x_p)^T$, para algún $p \in \mathbb{N}$.

   b) Se ejecuta un algoritmo de aprendizaje el cual utiliza las observaciones de entrenamiento $(\vec{x}, y)$ disponibles para ajustar los parámetros $\vec{w}$ y b tales que
   $$y \approx \vec{w}^T\vec{x} + b$$

   Esta forma de proceder simplifica el problema de estimar f, ya que sólo se deben estimar un conjunto de parámetros. Por lo general, el modelo que se elige para f no es el modelo matemático verdadero que la representa. Para solventar esta dificultad, se eligen modelos más flexibles: estos pueden ajustar diferentes formas de f pero se consideran más complejos de entender y deben estimar un gran número de parámetros. Ahora bien, usualmente, los modelos más flexibles suelen ocasionar el fenómeno conocido como sobreajuste[8] el cual se produce cuando el modelo brinda demasiada importancia a observaciones ruidosas, o errores, del conjunto de entrenamiento y, de esta forma, carecerá de una buena generalización.

2. En los métodos no paramétricos, en cambio, no se asume la forma de f: estos intentan estimarla, lo mejor posible, siguiendo las observaciones del conjunto de entrenamiento. Como consecuencia, en este caso, se necesita de un gran número de observaciones de entrenamiento.

2.2.2. Evaluación de modelos

Durante la presente tesina, sólo se considerará el problema del aprendizaje supervisado de un modelo clasificador. Es decir, escenarios en los cuales se tiene a disposición un conjunto de observaciones de entrenamiento $\mathscr{D} = \{(\vec{x}_i, y_i)\}_{i=1}^n$, donde y_i es una medición cualitativa, para todo i. Aplicando en $\mathscr{D}$ un algoritmo de aprendizaje supervisado, se aprende un modelo clasificador $\hat{f}$.

Una de las formas de evaluar el rendimiento del modelo aprendido es mediante la tasa de error de entrenamiento:

$$\epsilon_{entrenamiento} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i) = Ave(I(y_i \neq \hat{y}_i)), \tag{2.6}$$

[8]Del inglés, *overfitting*

donde $\hat{y}_i$ es la clase predicha para la i-ésima observación de $\mathscr{D}$ utilizando $\hat{f}$, y $I(y_i \neq \hat{y}_i)$ es igual a 1 si $y_i \neq \hat{y}_i$, e igual a 0 en caso contrario; es decir, la definición 2.6 calcula el promedio de las clasificaciones incorrectas realizadas por $\hat{f}$. Generalmente, no es interesante calcular el desempeño de $\hat{f}$ en el conjunto de entrenamiento, dado que un modelo con $\epsilon_{entrenamiento}$ cercano a cero suele estar sobreajustado a las observaciones de entrenamiento y, comúnmente, carecerá de una buena generalización (Hastie et al. 2009). Por lo tanto, lo que en realidad interesa es estimar el error de generalización del modelo aprendido: es decir, estimar la exactitud de las predicciones que se obtiene cuando el modelo es aplicado a datos no vistos durante entrenamiento (James et al. 2013). Una forma de estimar el error de generalización es calculando la tasa de error en un conjunto de ejemplos de prueba no visto por el método de aprendizaje durante la fase de entrenamiento. Es decir, dado un conjunto de pares (x^*, y^*) de observaciones de prueba no vistas durante la fase de aprendizaje, la tasa de error de prueba es el promedio:

$$\epsilon_{prueba} = Ave(I(y^* \neq \hat{y}^*))$$

En la práctica, el objetivo es aprender un modelo clasificador para el cual la tasa de error de prueba es pequeña. La Figura 2.2 muestra el comportamiento típico de $\epsilon_{entrenamiento}$ y ϵ_{prueba} en cuanto a la variación de la flexibilidad del modelo. El error de entrenamiento tiende a disminuir cada vez que se aumenta la complejidad del modelo; es decir, cada vez que se ajusta más la información: el modelo se adapta demasiado a las observaciones de entrenamiento y como resultado, por lo general, no generalizará bien (se dice que se incrementa la varianza y se reduce el sesgo). En cambio, si el modelo no es lo suficientemente complejo, provocará el fenómeno denominado subajuste, lo que nuevamente resultará en una generalización deficiente (se dice que se incrementa el sesgo y se reduce la varianza) (Hastie et al. 2009). Como ejemplo, en la Figura 2.3 se puede observar dos modelos de clasificación aprendidos donde uno de ellos aprendió a la perfección los patrones del conjunto de observaciones de entrenamiento, mientras que el otro aprendió un modelo más regularizado. Por lo general, en la práctica, se desea aprender un modelo no tan flexible ya que éstos comúnmente generalizan mejor.

En algunas situaciones se disponen de grandes conjuntos de prueba con el fin de estimar el error de generalización del modelo. El inconveniente sucede cuando lo anterior no sucede. Para afrontar la mencionada problemática, se suele utilizar el método de la *validación cruzada*[9] (CV).

Validación cruzada

La validación cruzada es un método de remuestreo para la estimación del error de prueba. En la ausencia de grandes conjuntos de ejemplos de prueba,

[9]Del inglés, *cross validation*

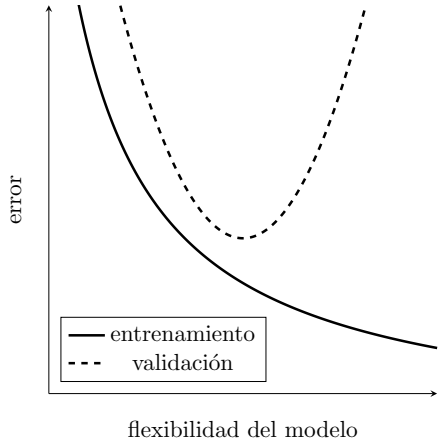

Figura 2.2: Comportamiento típico del error de prueba y entrenamiento en función de la flexibilidad del modelo.

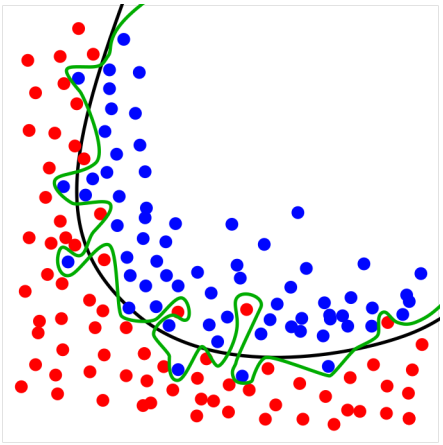

Figura 2.3: Ejemplo de dos modelos de clasificación aprendidos.

la CV utiliza las observaciones de entrenamiento para estimar la tasa del error. Existen diversos métodos de CV: *k-fold CV*[10] es uno de ellos. Como se explica en (Hastie et al. 2009), k-fold CV usa parte del conjunto de entrenamiento para entrenar el modelo y una parte distinta para probarlo. Precisamente, este método divide aleatoriamente el conjunto de entrenamiento en k folds de aproximadamente igual tamaño. El primer fold es asignado como el conjunto de validación y el método de aprendizaje es entrenado con los $k-1$ folds restantes. El error de prueba es calculado utilizando las observaciones en el fold apartado. Este procedimiento se repite k veces: en cada iteración, un grupo diferente de observaciones es tratado como el conjunto de validación o conjunto de prueba. Por lo tanto, el proceso devuelve k estimaciones del error

[10] *Fold* se traduce al español como *pliegue* o *grupo*

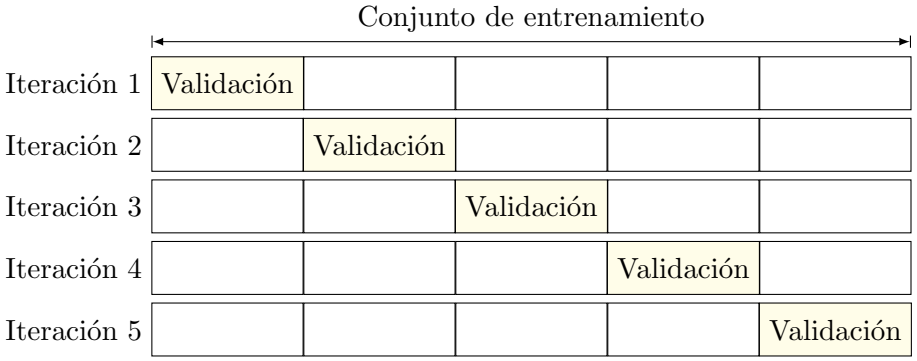

Figura 2.4: 5-fold CV.

de prueba; es decir, $\epsilon^{(1)}_{prueba}, \ldots, \epsilon^{(k)}_{prueba}$. Luego, la estimación de k-fold CV es el promedio de las anteriores mediciones; es decir:

$$CV^k_{\hat{f}} = \frac{1}{k}\sum_{i=1}^{k} \epsilon^{(i)}_{prueba}$$

La Figura 2.4 ilustra el método cuando $k = 5$. Como se detalla en (Hastie et al. 2009) y en (Kuhn et al. 2013), en la práctica es recomendable configurar $k = 5$ o $k = 10$.

Métricas

A continuación, se presentan algunas métricas importantes a la hora de evaluar un clasificador mediante un conjunto de observaciones de prueba. Sean $\mathbb{C}$ el conjunto de clases $\mathbb{C} = \{+1, -1\}$ y $\hat{f}$ un clasificador binario discreto[11], el rendimiento de $\hat{f}$ con respecto a un conjunto de prueba se registra típicamente en una matriz denominada *matriz de confusión* (Japkowicz et al. 2011) y se ilustra en la Figura 2.5. En la respectiva matriz, las mediciones *TP*, *FP*, *FN*, y *TN*[12] hacen referencia a los verdaderos positivos, los falsos positivos, los falsos negativos y los verdaderos negativos, respectivamente. Esta matriz es rica en información, dado que de la misma se pueden obtener otras métricas valiosas en el problema de la clasificación binaria (Tharwat 2018):

- *Exactitud* (ACC) es, probablemente, la métrica más utilizada para medir el desempeño de la clasificación. Esta es la fracción de predicciones que el modelo realizó correctamente, es decir:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = 1 - \epsilon_{prueba}$$

[11] Un clasificador discreto genera una etiqueta de clase por cada ejemplo de prueba (Japkowicz et al. 2011)

[12] Por *true positives*, *false positives*, *false negatives*, y *true negatives* en inglés, respectivamente

		Predicho	
		-1	$+1$
Real	-1	TN	FP
	$+1$	FN	TP

Figura 2.5: Matriz de confusión de $\hat{f}$

Esta métrica suele brindar resultados engañosos cuando ambas clases están significativamente desbalanceadas[13] (Borovicka et al. 2012).

- *Recall* (REC), o *sensibilidad*, es la fracción de ejemplos positivos que han sido correctamente clasificados:

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

- *Especificidad* es la fracción de ejemplos negativos que han sido correctamente clasificados:

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

- *Precision* (PREC) es la fracción de ejemplos recuperados que son positivos:

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

Ahora bien, existen los denominados clasificadores binarios continuos: estos retornan una probabilidad o un puntaje que representa el grado el cual una instancia de prueba es miembro de una clase. Estos valores pueden ser probabilidades que respetan teoremas estándares de probabilidad o pueden ser simplemente puntuaciones numéricas no calibradas. Un clasificador con la anterior propiedad puede ser utilizado junto a un umbral de decisión, o punto de operación, para producir un clasificador binario discreto (Japkowicz et al. 2011): si la respuesta del clasificador se encuentra por encima del umbral, el clasificador retorna $+1$, en otro caso -1. Existen métodos de análisis gráficos de evaluación para modelos clasificadores binarios continuos que permiten la visualización de rendimiento en un rango de umbrales de decisión. Una de las anteriores herramientas es la denominada curva ROC[14]. La curva ROC es un gráfico bidimensional en el cual el eje horizontal denota $1 - Especificidad$ (o la tasa de falsos positivos) y el eje vertical denota $Sensibilidad$ (o REC) del modelo clasificador. Por lo tanto, se dice que la mencionada curva estudia la

[13] Un conjunto de observaciones está *balanceado* cuando todas las clases están representadas con la misma proporción (Borovicka et al. 2012)

[14] Del inglés, *receiver operating characteristics*

relación entre la sensibilidad y la especificidad de un modelo para distintos umbrales de decisión (Japkowicz et al. 2011). Cada uno de estos puntos de operación tiene asociada una matriz de confusión que resume el rendimiento del clasificador en dicho punto. La selección del umbral de decisión depende del conocimiento del dominio como de objetivos de negocio específicos. La Fi-

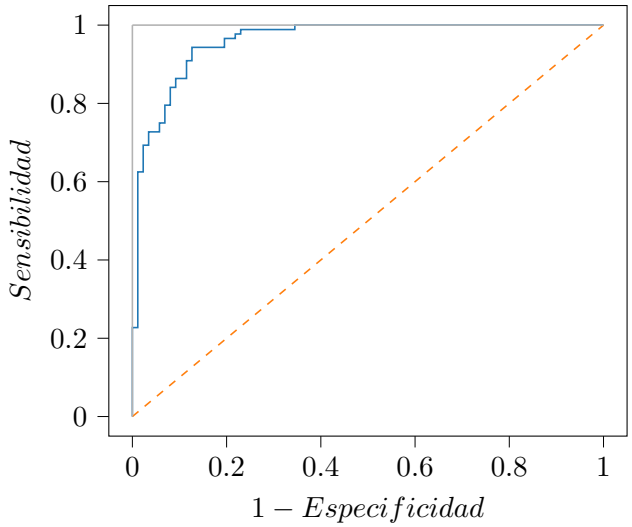

Figura 2.6: Ejemplo de curva ROC un modelo clasificador

gura 2.6 presenta un ejemplo de curva ROC de cierto modelo clasificador con respecto a un conjunto de prueba. Cualquier curva ROC generada mediante un conjunto finito de ejemplos de prueba es, en realidad, una función escalón, la cual se aproxima a una verdadera curva mientras el número de ejemplos de prueba se aproxima al infinito (Fawcett 2006). La curva gris de la Figura 2.6 muestra, además, el rendimiento de la clasificación perfecta; mientras que la linea punteada corresponde a la curva ROC de un clasificador que predice clases al azar. Como se explica en (Tharwat 2018), cualquier clasificador en el triángulo inferior derecho posee peor rendimiento que cualquier clasificador en el triángulo superior izquierdo.

2.3. Sistemas de recomendación

Los sistemas recomendadores son herramientas que proporcionan sugerencias de ítems potencialmente interesantes para un usuario en particular (Ricci et al. 2015). Para realizar esta tarea, estos sistemas recopilan información de los usuarios con respecto a sus preferencias y esta información es analizada con el fin de estimar sus intereses ante distintos ítems. La recopilación de información de preferencia de los usuarios puede ser obtenida de forma explícita

(recolectando opiniones de los usuarios) o de forma implícita (por ejemplo, observando el comportamiento de los mismos) (Ricci et al. 2015).

Los sistemas de recomendación ayudan a las personas en la toma de decisiones en diversos escenarios. Son herramientas de suma importancia ya que con el incremento de la información en la web en la actualidad, un análisis constante de toda nueva información es prácticamente imposible para el ser humano. El desarrollo de estos sistemas involucra la intersección de muchas disciplinas: inteligencia artificial, interacción humano-máquina, minado de datos, estadística, aprendizaje automatizado y ciencias cognitivas (Ricci et al. 2015). Los sistemas recomendadores se basan en la idea de que solemos considerar las recomendaciones proporcionadas por otros para tomar decisiones rutinarias (Shardanand et al. 1995). Actualmente, los sistemas de recomendación son empleados para resolver diversas problemáticas de cualquier aspecto de la vida moderna, tales como viajes, música, películas y libros. Cada día más empresas se dedican a desarrollar sus propios sistemas recomendadores para brindar un mejor servicio a sus usuarios, lo cual les otorga una ventaja sobre la competencia. Entre los casos de éxitos mas populares se pueden nombrar: Netflix[15] y Amazon[16].

2.3.1. Modelos de recomendación

En la literatura, existen numerosos modelos o técnicas de recomendación de ítems. En esta tesina se introducen tres técnicas.

Los modelos básicos de sistemas de recomendación, comúnmente, trabajan con dos tipos de datos (Aggarwal 2016):

(a) Las interacciones usuario-item (tales como *ratings*[17] o comportamientos de compra), y

(b) características de usuarios e ítems.

Los modelos que utilizan (a) son referidos como métodos basados en *filtrado colaborativo*, mientras los sistemas que utilizan (b) son denominados *basados en contenido*. Los sistemas que dependen de fuentes de conocimiento distintas a las enunciadas se conocen como *basados en conocimiento*.

Basados en filtrado colaborativo

El filtrado colaborativo es el proceso de filtrar o evaluar ítems mediante las opiniones de otras personas (Schafer et al. 2007). En la web se puede descubrir no sólo la opinión de un conjunto reducido de personas, sino de cientos

[15] `https://netflix.com`

[16] `https://amazon.com`

[17] *Valoraciones*, en español. Estas, por lo general, pertenecen al intervalo $[0, 5]$ de los números naturales, o son valoraciones categóricas como “relevante” o “no relevante”

de miles de ellas. La velocidad de las computadoras permite analizar opiniones prácticamente en tiempo real. La idea fundamental de todo sistema de recomendación basado en filtrado colaborativo consiste en que la valoración de cierto usuario con respecto a un ítem en particular es similar a la valoración que le brindó otro usuario si ambos usuarios, en el pasado, han valorado otros ítems de forma similar (Ricci et al. 2015). A diferencia de los sistemas basados en contenido, estos sistemas utilizan la premisa de que las personas con gustos similares valorarán nuevos ítems de manera similar. Mientras que los sistemas de recomendación basados en contenido suponen que ítems con similares características serán valorados de forma similar. En los sistemas basados en filtrado colaborativo, la similitud entre usuarios es utilizada para predecir calificaciones faltantes en la matriz usuario-item (comúnmente denominada matriz de utilidad).

Existen dos tipos de métodos utilizados en la presente técnica de recuperación (Aggarwal 2016):

- **Basados en memoria**. Estos métodos, también denominados basados en vecinos, se consideran los más comunes a la hora de implementar un sistema basado en filtrado colaborativo. Esta estrategia utiliza la matriz usuario-item para calcular similitudes entre usuarios o ítems. Esta metodología se diversifica en las siguientes dos modalidades. Sea u un usuario:

  - *Filtrado colaborativo basado en usuarios.* Se procede a encontrar un conjunto S_u de usuarios similares a u y aproximar calificaciones faltantes del mencionado usuario hacia cierto ítem i computando el promedio de las calificaciones de los usuarios de S_u con respecto a i.
  - *Filtrado colaborativo basado en ítems.* Con el objetivo de predecir la calificación de u hacia i se determina un conjunto S_i^u de ítems similares a i, los cuales el mencionado usuario ya ha calificado en el pasado, y se procede a computar el promedio entre todas las calificaciones de los ítems en S_i^u para obtener una aproximación de la calificación de u hacia el ítem i.

- **Basados en modelos**. Aquí se implementan diversos modelos de aprendizaje automatizado o minado de datos con el objetivo de predecir la calificación de ítems aún no calificados. Una de las técnicas más utilizadas es la indexación semántica latente.

Basados en contenido

Los sistemas recomendadores basados en contenido recomiendan ítems a cierto usuario en base a la descripción de los respectivos ítems y en base al perfil de intereses del usuario (Pazzani et al. 2007). Si u es un usuario que

utiliza un sistema que implementa esta técnica de recomendación, el sistema analiza un conjunto de descripciones de ítems previamente evaluados por u y construye su perfil. Este perfil se considera una representación de intereses de u y se utiliza para recomendar nuevos ítems potencialmente relevantes para el usuario en cuestión (Aggarwal 2016). Por lo general, un perfil de usuario se conforma mediante:

(a) Un conjunto de $n \in \mathbb{N}$ observaciones de entrenamiento

$$\mathscr{D} = \{(descr(i_1), val(i_1)), \ldots, (descr(i_n), val(i_n))\},$$

donde $descr(i_j)$ y $val(i_j)$ es la descripción y la valuación (cuantitativa o categórica) del ítem j, respectivamente; junto con

(b) un modelo de clasificación o regresión aprendido mediante $\mathscr{D}$.

En estos modelos no es necesario la experiencia de otros usuarios, como sí sucede en los basados en filtrado colaborativo. Los basados en contenido se enfocan en (i) la información de relevancia que el usuario brindó a cierto conjunto de ítems a lo largo del tiempo y (ii) en los atributos de estos últimos.

Los modelos basados en contenido son utilizados, comúnmente, en situaciones en las cuales se puede representar cada ítem mediante un gran número de atributos. Por lo general, los atributos son términos que se extraen de la descripción textual de los ítems. Por consecuencia, los sistemas de recomendación basados en contenido son utilizados en dominios ricos en información textual. Un ejemplo clásico es la recomendación de artículos de noticias (Kompan et al. 2010). En este trabajo, la clasificación de texto es de suma importancia y se introduce en el Capítulo 3.

Como se explica en (Aggarwal 2016), los principales componentes de un sistema basado en contenido incluyen una parte de preprocesamiento y extracción de atributos (offline), una parte de aprendizaje (offline) y la parte de predicción de respuestas en tiempo real. Aquí, las partes que se consideran offline se utilizan para aprender un modelo de clasificación o regresión. Luego, este modelo se utiliza para la generación en línea de recomendaciones para usuarios. Es posible resumir cada etapa como sigue:

1. **Preprocesamiento y extracción de atributos**: de cada ítem se extrae un conjunto de atributos con el objetivo de caracterizarlo lo mejor posible. Este paso se encarga de transformar todo ítem en su representación vectorial y es el primer paso que debe realizarse en un sistema de recomendación.

2. **Aprendizaje del perfil de usuario**: se debe construir un modelo con el objetivo de predecir, lo más fielmente, el interés del usuario acerca de cualquier ítem para los cuales este último aún no ha calificado. Estos modelos se basan en el historial de intereses del usuario. Si el concepto de

relevancia utilizado fue cuantitativo, se aprende un modelo de regresión; mientras que en el caso contrario se aprende un modelo de clasificación. El modelo aprendido, usualmente, se denomina *perfil del usuario* porque relaciona intereses del respectivo usuario con características de ítems.

3. **Recomendación de ítems**: este paso se considera el último en aplicar en un sistema recomendador. Consiste en utilizar el modelo aprendido para predecir la calificación del usuario con respecto a un nuevo item.

Por lo general, los sistemas recomendadores basados en contenido se emplean en situaciones en donde (i) no hay demasiada información de validación de usuarios con respecto a ítems y por lo tanto, no es viable considerar implementar un sistema de recomendación basado en filtrado colaborativo; y (ii) cuando se puede extraer un número significativo de características de los ítems. Se considera que estos modelos afrontan de forma parcial el problema del arranque en frío[18] cuando el número de usuarios en el sistema es relativamente pequeño. Pero, por otro lado, el mencionado problema ocurre cuando se presentan nuevos usuarios en el sistema. Utilizar sistemas de recomendación basados en contenido en el caso anterior, por lo general, brinda pobres resultados ya que al comienzo se carece de largos historiales de intereses de los usuarios. Por lo tanto, se dice que los sistemas recomendadores basados en contenido pueden afrontar el problema del arranque en frío con respecto a nuevos ítems, pero no son capaces de hacerlo con respecto a nuevos usuarios.

Hasta aquí, se ha presentado una técnica de recomendación con la cual aprender modelos específicos para un usuario en base a sus intereses y de esta manera, llevar a cabo la tarea de recomendación. El modelo aprendido será capaz de estimar la calificación del usuario ante nuevos ítems. Se nota que ningún sistema recomendador basado en contenido puede brindar buenas recomendaciones si el historial de intereses del usuario no contiene suficiente información (Pazzani et al. 2007). En estos casos, los sistemas basados en filtrado colaborativo tienden a brindar mejores resultados (Schafer et al. 2007).

Ahora bien, si γ es el perfil del usuario aprendido, luego:

$$S \longrightarrow \gamma \longrightarrow S_R,$$

donde S es un conjunto de ítems y $S_R \subseteq S$ es el conjunto de aquellos ítems potencialmente relevantes para el usuario.

Los sistemas recomendadores son capaces de recolectar opiniones de los usuarios acerca de los ítems que son recomendados. Por lo tanto, el usuario suele brindar su crítica o feedback, de forma explícita y categórica, acerca de cada ítem recomendado por el sistema:

$$S \longrightarrow \gamma \longrightarrow S_R \underset{feedback}{\rightsquigarrow} \checkmark \text{o } \times$$

[18]En un sistema recomendador, el problema del arranque en frío está relacionado con la escasez de información para usuarios y elementos disponibles en el algoritmo de recomendación (Lika et al. 2014)

Con esta información recolectada, eventualmente, se aprende un perfil del usuario más robusto.

Basados en conocimiento

Las técnicas de recomendación anteriores son, por lo general, las más aplicadas en un sistema de recomendación. Ahora bien, existen casos en los cuales no se posee largos historiales de preferencias de usuarios con respecto a ítems, o interacciones usuario-item como sucede en los sistemas basado en contenido y filtrado colaborativo, respectivamente. Los sistemas *basados en conocimiento* asisten al modelado del perfil del usuario utilizando otras fuentes de conocimiento distintas a las anteriormente nombradas (Felfernig et al. 2008). Esta técnica de recomendación es usualmente utilizada en situaciones donde el dominio en el cual yacen los ítems es complejo; por ejemplo, la compra de autos y bienes raíces.

Los sistemas basados en conocimiento solicitan explícitamente los requerimientos de los usuarios mediante una interacción típicamente modelada en forma de diálogo (Aggarwal 2016). Como se ha comentado, el dominio de los ítems suele ser complejo; por lo tanto, resulta difícil para los usuarios enunciar o incluso comprender cómo sus requisitos coinciden con la disponibilidad de los ítems. Para intentar solventar la anterior dificultad, durante el dialogo que se establece entre el sistema y el usuario, el sistema utiliza bases de conocimiento que describen el dominio de los ítems con el propósito de intentar facilitar la recuperación y exploración de requerimientos.

Capítulo 3

Introducción a la clasificación de texto

El presente capítulo se enfoca en una de las herramientas más importantes del procesamiento del lenguaje natural (PLN): la clasificación de texto (CT). La CT, las nociones de los sistemas de información de texto y los conceptos de los sistemas de recomendación son fundamentales para el sistema que se propone en esta tesina para enfrentar la problemática de la matricería legal.

Durante el desarrollo de los capítulos anteriores, se ha comentado la importancia de la información textual en la cotidianidad, principalmente en la web. Además, se ha destacado que si bien el texto puede ser extremadamente rico en información, la naturaleza no estructurada del mismo dificulta la extracción de conocimiento. Por lo tanto, analizar texto de forma artificial se ha convertido en uno de los dos componentes más importantes en un sistema de información de texto (el componente restante es el acceso a la información) y uno de los instrumentos clave en el análisis de texto es su clasificación.

El PLN aún no ha llegado al punto en el cual automáticamente logre entender documentos de texto en su totalidad. La CT, como herramienta, no escapa del problema anterior. Con lo cual, aprender a clasificar documentos de forma artificial, por lo general, requiere de la ayuda explícita o implícita del humano.

A continuación, se define el problema de la clasificación de texto. Particularmente, el presente capítulo introduce uno de los algoritmos de aprendizaje más utilizado en la CT: el clasificador de vectores soporte.

3.1. El problema de la clasificación de texto

En las últimas décadas, el problema de la CT ha sido foco de estudio en el minado de datos, aprendizaje automatizado, y recuperación de información (Aggarwal et al. 2012). En la actualidad, la CT se aplica en diversas circunstancias: detección automática de noticias falsas (Pérez-Rosas et al. 2017),

análisis de sentimiento de usuarios con respecto a películas, libros y hoteles (Hussein 2018) y el filtrado de spam en casillas de correo electrónico (Zhang et al. 2007), entre otras numerosas aplicaciones.

Como se explica en (Sebastiani 2002), la CT es la actividad de etiquetar documentos de texto escritos en lenguaje natural con categorías temáticas pertenecientes a un conjunto predefinido. En (Manning et al. 2008), la CT se define formalmente como sigue. Sean $d \in \mathbb{X}$ la descripción de un documento de texto, $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ un conjunto predefinido de J clases y $\mathscr{D}$ un conjunto de entrenamiento de pares $(d, c) \in \mathbb{X} \times \mathbb{C}$. Luego, mediante la información que se dispone en el conjunto $\mathscr{D}$ y utilizando algún algoritmo de aprendizaje, se aprende un clasificador γ que mapea documentos a clases; es decir:

$$\gamma : \mathbb{X} \to \mathbb{C}$$

Como se observa, la CT es una técnica de aprendizaje supervisado: si Γ es un algoritmo de aprendizaje supervisado, luego $\gamma = \Gamma(\mathscr{D})$. Si d es un documento el cual se desconoce su clase, se procede a utilizar el clasificador aprendido para predecirla; es decir: $\gamma(d) = c$, para algún $c \in \mathbb{C}$.

La definición de CT presentada indica que todo documento es miembro de, exactamente, una sola clase. Este tipo de problema se denomina clasificación *multi-clase* (Manning et al. 2008). Por lo general, en la recuperación de información, se utiliza la clasificación *multi-valor*: aquí, cada documento puede pertenecer a una sola clase, a varias clases de forma simultánea o a ninguna. A partir de este momento, sólo se considera el problema de clasificación multi-clase. Más adelante en el trabajo, se hace foco en el escenario multi-valor, indispensable para el modelado del sistema que se propone en esta tesina.

A lo largo del presente capítulo se estudia la CT en el modelo Espacio-Vectorial; particularmente, modelos de clasificación lineal: dada la naturaleza propia del texto, en esta tesina se detalla que los modelos de clasificación lineal son, por lo general, los que brindan mejores resultados a la hora de la clasificación de texto. Si bien, en este trabajo, se considera que la CT en el MEV es la más importante de introducir, en la literatura existen otros modelos de clasificación que se consideran apropiados para la CT como, por ejemplo, los clasificadores basados en reglas y el clasificador de k vecinos más cercanos (Aggarwal 2018).

3.2. Clasificación en el modelo Espacio-Vectorial

A lo largo de esta sección se utilizan conceptos del MEV. En este último, como se ha detallado anteriormente, todo documento d, luego de su respectivo preprocesamiento de texto, tiene su representación vectorial $\vec{v}_d$ mediante la adopción del esquema de pesaje tf-idf; es decir:

$$\vec{v}_d = (tf\text{-}idf_{t_1,d}, tf\text{-}idf_{t_2,d}, \ldots, tf\text{-}idf_{t_{|V|},d}) \in \mathbb{R}_0^{+^{|V|}},$$

donde V es un vocabulario de términos.

Dado que la representación del texto en el MEV se encuentra en un espacio de alta dimensionalidad, los modelos lineales de clasificación se consideran especialmente atractivos a la hora de la CT (Aggarwal 2018) (Yuan et al. 2012) (Pilászy 2005).

3.2.1. Clasificación lineal

A lo largo de esta sección, con el propósito de simplificar el problema de la CT, sólo se considerará el problema de la clasificación binaria, donde el conjunto de clases se define como $\mathbb{C} = \{+1, -1\}$. Más adelante, se describe cómo utilizar clasificadores binarios en un escenario multi-valor, cuando $|\mathbb{C}| > 2$. Con esta simplificación, en (Manning et al. 2008) se define un *clasificador lineal* como un clasificador binario γ que decide la membresía de clase de un ejemplo de prueba mediante la comparación de una combinación lineal de las características del ejemplo con un umbral de decisión:

$$\gamma(\vec{x}) = signo(\vec{w}^T\vec{x} + b) = \begin{cases} +1 & \text{si } \vec{w}^T\vec{x} + b > 0 \\ -1 & \text{en otro caso} \end{cases} \tag{3.1}$$

En otras palabras, un clasificador lineal es una función de la forma

$$\vec{x} \mapsto signo(\vec{w}^T\vec{x} + b)$$

que etiqueta positivamente todos los puntos que yacen en un lado del hiperplano $\vec{w}^T\vec{x} + b = 0$ y negativamente a todos los demás[1] (Mohri et al. 2018). Al hiperplano que se utiliza en la definición 3.1, se lo denomina *hiperplano de separación* o *hiperplano de decisión* (Hastie et al. 2009): los parámetros $\vec{w}$ y b se aprenden mediante las observaciones en el conjunto de entrenamiento. En la literatura, existen diferentes algoritmos que ajustan estos parámetros y en la presente sección se introducen algunos de ellos.

Ahora, sea $\mathscr{D}$ un conjunto de observaciones de entrenamiento, si existe un hiperplano que separa perfectamente las dos clases, entonces se dice que ambas clases son *linealmente separables* (Manning et al. 2008). De hecho, si la separabilidad lineal se satisface, existen infinitos hiperplanos que separan ambas clases. En la Figura 3.1 se observa un ejemplo de dos clases linealmente separables. Usualmente, la CT se considera un problema lineal: con el incremento de la dimensionalidad, la probabilidad de separar linealmente ambas clases incrementa (Manning et al. 2008) (Aggarwal 2018).

En la presente tesina, como metodología, primero se presentará el escenario en que toda observación de un conjunto de entrenamiento puede ser perfectamente clasificada por un clasificador lineal para, más tarde, introducir el problema general en el cual ambas clases son linealmente cuasi-separables.

[1] En un espacio p-dimensional, un *hiperplano* $\mathscr{H} : \vec{w}^T\vec{x} + b = 0$ es un subespacio afín de dimensión $p - 1$ (James et al. 2013). $\vec{w}$, denominado vector *normal*, regula la orientación de $\mathscr{H}$ y b es un escalar que regula la distancia del mismo hacia el origen

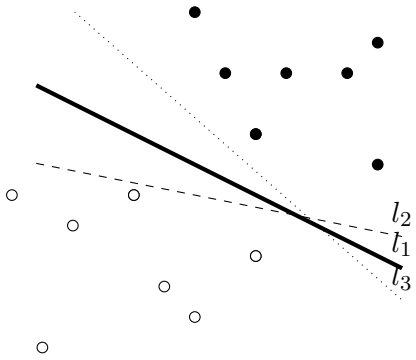

Figura 3.1: Ejemplo de dos clases linealmente separables. l_1, l_2, y l_3 son ejemplos de hiperplanos de decisión que separan ambas clases.

Como se ha comentado, cuando ambas clases son linealmente separables, se necesita un criterio para elegir uno de los infinitos hiperplanos de separación. El denominado *clasificador de vectores soporte* propone uno de estos criterios y es considerado, en la literatura, como uno de los algoritmos de aprendizaje con mejores resultados en la CT (Aggarwal 2015) (Aggarwal et al. 2012) (Manning et al. 2008) (Joachims 2001).

Clasificador de vectores soporte

Sea $\mathscr{D}$ un conjunto de observaciones de entrenamiento linealmente separable definido como

$$\mathscr{D} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, +1\}\}_{i=1}^{n}.$$

Luego, tal como se comentó, se necesita un criterio para seleccionar uno de los infinitos hiperplanos de separación. Una de las opciones más aceptadas en la literatura es elegir el hiperplano de decisión que se encuentre más alejado de las observaciones de entrenamiento y se conoce como *hiperplano separador de margen máximo* o *hiperplano de separación óptimo*. En otras palabras, el hiperplano de separación de margen máximo es aquel que separa las dos clases y maximiza la distancia (perpendicular) al punto más cercano desde cualquier clase (Hastie et al. 2009). A continuación, se describe cómo encontrar dicho hiperplano.

Dado que $\mathscr{D}$ es linealmente separable, existe un hiperplano $\mathscr{H} : \vec{w}^T\vec{x} + b = 0$ que separa perfectamente ambas clases. Ahora, con respecto a $\mathscr{H}$, es posible construir dos hiperplanos paralelos, $\mathscr{H}_{-1}$ y $\mathscr{H}_{+1}$, que toquen los datos de entrenamiento de clases opuestas a cada lado, y entre ellos no existan observaciones de entrenamiento. $\mathscr{H}_{-1}$ y $\mathscr{H}_{+1}$ se denominan *hiperplanos marginales* y a las observaciones que yacen en cada uno de ellos se las llama *vectores soporte*. La distancia entre los hiperplanos marginales es denominada *margen* y $\mathscr{H}$, el hiperplano de decisión, se encuentra exactamente en el medio de los

dos hiperplanos marginales para lograr la clasificación más precisa (Aggarwal 2015). Ahora bien, los coeficientes $\vec{w} = (w_1, \ldots, w_p)^T$ y b del hiperplano $\mathscr{H}$ deben ser ajustados mediante las observaciones de entrenamiento con el objeto de maximizar el mencionado margen. Los anteriores coeficientes se hallan mediante la resolución de un problema de optimización, el cual se describe y formula como sigue.

Primeramente, dado que $\mathscr{H}$ separa linealmente ambas clases, se satisfacen las siguientes restricciones:

$$\vec{w}^T\vec{x}_i + b > 0 \ \forall i : y_i = +1$$
$$\vec{w}^T\vec{x}_i + b \leq 0 \ \forall i : y_i = -1$$

Es decir, todas las observaciones de entrenamiento etiquetadas con la clase positiva se encuentran de un lado de $\mathscr{H}$, mientras que los datos de entrenamiento de clase negativa se encuentran del otro lado del hiperplano. Como se ha comentado anteriormente, se asume que $\mathscr{H}$ se encuentra exactamente en el medio de los hiperplanos marginales $\mathscr{H}_{+1}$ y $\mathscr{H}_{-1}$, los cuales son paralelos al mencionado hiperplano de separación y pasan por las observaciones de entrenamiento más cercanas de las clases positivas y negativas. Luego, las ecuaciones de los hiperplanos marginales son:

$$\mathscr{H}_{+1} : \vec{w}^T\vec{x} + b = +1$$
$$\mathscr{H}_{-1} : \vec{w}^T\vec{x} + b = -1$$

En los dos hiperplanos anteriores yacen los vectores soporte de cada clase, respectivamente. Se asume, además, que no existen observaciones de entrenamiento en la región de decisión que se forma entre los mencionados hiperplanos marginales y que todas las observaciones de entrenamiento para cada clase son mapeados a una de las dos regiones extremas. Estas restricciones, denominadas *restricciones del margen*, se formulan como sigue:

$$\vec{w}^T\vec{x}_i + b \geq +1 \ \forall i : y_i = +1 \tag{3.2a}$$
$$\vec{w}^T\vec{x}_i + b \leq -1 \ \forall i : y_i = -1 \tag{3.2b}$$

Las restricciones 3.2 se pueden formular en una única restricción observando las siguientes particularidades:

$$\begin{aligned} & \vec{w}^T\vec{x}_i + b \leq -1 \\ \iff & y_i(\vec{w}^T\vec{x}_i + b) \geq y_i(-1) \\ \iff & y_i(\vec{w}^T\vec{x}_i + b) \geq +1 \ \forall i : y_i = -1 \end{aligned} \tag{3.3}$$

$$\begin{aligned} & \vec{w}^T\vec{x}_i + b \geq 1 \\ \iff & y_i(\vec{w}^T\vec{x}_i + b) \geq y_i(+1) \\ \iff & y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \ \forall i : y_i = 1 \end{aligned} \tag{3.4}$$

Por lo tanto, combinando 3.3 y 3.4 se tiene que:

$$y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \ \forall i : 1 \leq i \leq n$$

Ahora bien, el objetivo, entonces, es maximizar la distancia entre los hiperplanos $\mathscr{H}_{-1}$ y $\mathscr{H}_{+1}$, ya que la distancia entre ellos es lo que se define como margen. Se puede demostrar que la distancia entre ambos hiperplanos marginales es $\frac{2}{\|\vec{w}\|}$; por lo tanto, el problema de la maximización del margen se reduce a solucionar el siguiente problema de optimización[2]:

$$\begin{aligned} \underset{\vec{w},b}{\text{maximizar}} \quad & \frac{2}{\|\vec{w}\|} \\ \text{sujeto a} \quad & y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \ \forall i : 1 \leq i \leq n \end{aligned} \tag{3.5}$$

Dado que maximizar $\frac{2}{\|\vec{w}\|}$ es equivalente a minimizar $\frac{1}{2}\|\vec{w}\|^2$, el problema de la maximización del margen se transforma en un problema de minimización, conocido como problema de optimización primal (Mohri et al. 2018):

$$\begin{aligned} \underset{\vec{w},b}{\text{minimizar}} \quad & \frac{1}{2}\vec{w}^T\vec{w} \\ \text{sujeto a} \quad & y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \ \forall i : 1 \leq i \leq n \end{aligned} \tag{3.6}$$

La ventaja de 3.6 con respecto a 3.5 es que en la primera se optimiza una función cuadrática sujeto a restricciones lineales y se puede resolver mediante la programación cuadrática, concepto fuera del alcance de esta tesina.

Una vez resuelto el problema de optimización especificado en 3.6, el clasificador de margen máximo queda definido mediante el hiperplano $\mathscr{H}$; es decir:

$$\gamma(\vec{x}) = signo(\vec{w}^T\vec{x} + b)$$

En la Figura 3.2 se ilustra un ejemplo de clasificador de margen máximo en donde se observan, exactamente, tres vectores soporte. Como se explica en (James et al. 2013), el hiperplano de margen máximo depende directamente de los vectores soporte y no de los demás: cualquier modificación de los vectores soporte implica un cambio en el hiperplano de decisión de margen máximo, mientras que las modificaciones en las demás observaciones no impactan en la definición de dicho hiperplano.

Anteriormente se ha comentado que el problema de la CT se considera un problema lineal. Ahora bien, en la práctica, usualmente las observaciones del conjunto de entrenamiento son linealmente cuasi-separables: observaciones tales como aquellas mal etiquetadas o valores atípicos violan la separación lineal. Una manera de afrontar el anterior escenario es maximizar el margen pero permitir que algunas observaciones de entrenamiento se encuentren del lado incorrecto del mismo. Para esto, existe una extensión del clasificador de

[2] $\|\cdot\|$ es la norma euclídea

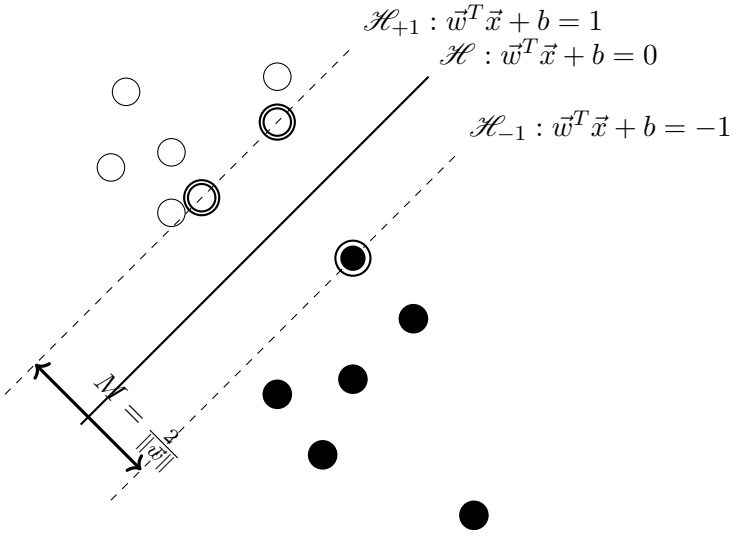

Figura 3.2: Ejemplo de clasificador de margen máximo. $\mathscr{H}$ es el hiperplano serparador de margen máximo. Los hiperplanos marginales, $\mathscr{H}_{+1}$ y $\mathscr{H}_{-1}$, son representados mediante lineas punteadas. Se observan exactamente tres vectores soporte.

margen máximo denominada *clasificador de vectores soporte*, o *clasificación de margen blando* (Hastie et al. 2009) la cual se presenta a continuación.

En la clasificación de margen blando la noción de margen se transforma en una más liviana porque se permite que las observaciones de entrenamiento violen las restricciones del margen a expensas de una penalización (Aggarwal 2015). Dada una observación de entrenamiento $\vec{x}_i$, se cuantifica su respectiva violación de margen mediante la variable $\xi_i \geq 0$. La variable[3] ξ_i representa la distancia (perpendicular) de la observación de entrenamiento $\vec{x}_i$ a su correspondiente hiperplano de separación cuando $\vec{x}_i$ se encuentra en el lado incorrecto de este último: es decir, ξ_i es la distancia por la que $\vec{x}_i$ sobrepasa la desigualdad deseada $y_i(\vec{w}^T\vec{x}_i + b) \geq 1$. La Figura 3.3 ilustra la situación anterior. Los valores de estas variables son iguales a cero cuando las respectivas observaciones de entrenamiento se encuentran en el lado correcto de los hiperplanos marginales. Luego, las nuevas restricciones de margen se formulan de la siguiente forma:

$$\vec{w}^T\vec{x}_i + b \geq +1 - \xi_i \ \forall i : y_i = +1$$
$$\vec{w}^T\vec{x}_i + b \leq -1 + \xi_i \ \forall i : y_i = -1,$$

donde $\xi_i \geq 0 \ \forall i : 1 \leq i \leq n$. Las anteriores restricciones se pueden escribir en una única restricción:

$$y_i(\vec{w}^T\vec{x}_i + b) \geq 1 - \xi_i \ \forall i : 1 \leq i \leq n, \tag{3.7}$$

[3]Las variables ξ se suelen llamar variables *slack*

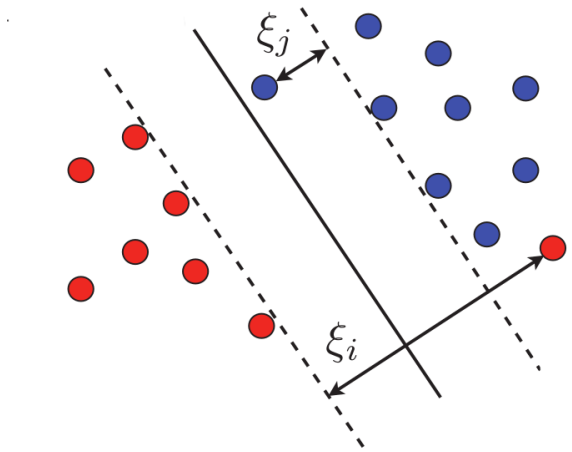

Figura 3.3: Ejemplo de un clasificador de vectores soporte. El hiperplano separador, o hiperplano de decisión, clasifica incorrectamente a la observación $\vec{x_i}$ y correctamente a $\vec{x_j}$ (aunque con margen menor a 1).

donde $\xi_i \geq 0 \; \forall i : 1 \leq i \leq n$.

Las violaciones de márgenes son penalizadas por un hiperparámetro[4] de regularización C. Luego, el objetivo final es obtener un margen lo más largo posible y, al mismo tiempo, mantener el número de observaciones con $\xi > 0$ lo más pequeño posible. Por lo tanto, el hiperplano que define el clasificador de vectores soporte es la solución del siguiente problema de optimización:

$$\begin{aligned} \underset{\vec{w},b,\vec{\xi}}{\text{minimizar}} \quad & \frac{1}{2}\vec{w}^T\vec{w} + C\sum_i \xi_i \\ \text{sujeto a} \quad & \xi_i \geq 0, \; y_i(\vec{w}^T\vec{x}_i + b) \geq 1 - \xi_i \; \forall i : 1 \leq i \leq n \end{aligned} \tag{3.8}$$

Configurar pequeños valores de C resultan en márgenes más holgados (mayor regularización), mientras que configurar grandes valores resultan en márgenes más angostos (menor regularización) y se tendrá un comportamiento similar al clasificador de margen máximo[5]. Se dice que el hiperparámetro C determina la compensación, o tradeoff, entre la maximización del margen y la minimización del error ($\sum_i \xi_i$), y su valor es típicamente configurado mediante validación cruzada (Alpaydin 2010).

[4]Un *hiperparámetro* es un parámetro cuyo valor no puede estimarse directamente a partir de las observaciones de entrenamiento (Kuhn et al. 2013). Estos deben ser configurados antes de comenzar el proceso de aprendizaje y existen diversas técnicas para encontrar buenos valores

[5]Como se detalla en (Hastie et al. 2009), el caso separable corresponde cuando $C = \infty$

Como se explica en (James et al. 2013), si $\vec{x^*}$ es un ejemplo de prueba, la distancia con signo $f(\vec{x}) = \vec{w}^T\vec{x} + b$ indica el grado de confianza de la asignación de clase para $\vec{x^*}$: es decir, si $f(\vec{x^*})$ es un valor lejano al cero, la confianza acerca de la asignación de clase para $\vec{x^*}$ aumenta; en cambio, si $f(\vec{x^*})$ es cercano al cero, entonces la observación se encuentra cercana al hiperplano de decisión y la certeza acerca de la asignación de clase para $\vec{x^*}$ decrece. En la literatura, se dice que el umbral de decisión por defecto del clasificador de vectores soporte es el valor cero (Shanahan et al. 2003) (Yu et al. 2010). Ahora bien, como se detalla en (Lin et al. 2013), cuando se posee un conjunto de observaciones de entrenamiento con clases significativamente desbalanceadas el clasificador de vectores soporte tiende a favorecer a la clase mayoritaria, resultando en una pobre exactitud (ACC) en la predicción de la clase minoritaria. Para solventar lo anterior, existen dos principales técnicas de corrección para mejorar el rendimiento del clasificador (Chen et al. 2006):

- **Técnicas de muestreo**. Estas son las más comunes de emplear en la práctica y según (Lin et al. 2013) se considera la estrategia de corrección más utilizada con respecto al clasificador de vectores soporte. El objetivo de estos métodos es cambiar las distribuciones de clases aumentando el número de observaciones de la clase minoritaria (sobremuestreo) o disminuyendo el número de observaciones de la clase mayoritaria (submuestreo). Estas técnicas no modifican el algoritmo de clasificación.

- **Ajuste del umbral de decisión**. Como se describe en (Lin et al. 2013), usar el umbral de decisión por defecto en el clasificador de vectores soporte en un problema de clases significativamente desbalanceadas producirá una medición alta con respecto a la exactitud en la predicción de la clase mayoritaria y una medición baja con respecto a la exactitud en la predicción de la clase minoritaria. La técnica de ajuste del umbral de decisión, como se explica en (Chen et al. 2006), se realiza mediante el análisis de la curva ROC y modifica el umbral de decisión por defecto del algoritmo de clasificación con el propósito de aumentar sensibilidad y decrementar especificidad, o viceversa, sujeto a condiciones u objetivos de estudio específicos.

Hasta aquí, se ha visto que el clasificador de vectores soporte es una extensión del clasificador de margen máximo. En la práctica, aunque $\mathscr{D}$ sea linealmente separable, se recomienda aprender un hiperplano de margen blando dado que el clasificador de margen máximo suele sobreajustar las observaciones de entrenamiento y provocar sensibilidad en cuanto a observaciones individuales (James et al. 2013).

Gracias a la naturaleza del texto en el modelo Espacio-Vectorial y al hiperparámetro de regularización del clasificador de vectores soporte, este último es sumamente atractivo de emplear en la CT, dado que se prefiere una solución

que separe la mayor parte de las observaciones mientras ignora algunos documentos ruidosos (Manning et al. 2008). Ahora bien, cuando se aprende un clasificador de texto, por lo general, se poseen muchas menos observaciones de entrenamiento que dimensiones en el espacio vectorial. Particularmente, en (Joachims 2002) se explica cómo el clasificador de vectores soporte puede evitar la "maldición de dimensionalidad" en la CT incluso sin aplicar un paso de selección de características, lo cual es esencial para muchos métodos convencionales.

El clasificador de margen máximo y el clasificador de vectores soporte pertenecen a un conjunto de algoritmos de aprendizaje llamado *máquinas de vectores soporte* (SVM, del inglés *Support Vector Machines*) (Hastie et al. 2009). En especial, la *máquina de vectores soporte* es una extensión del clasificador de vectores soporte para problemas de clasificación no lineales. Aunque el desarrollo de la máquina de vectores soporte queda fuera del alcance de la presente tesina, este método utiliza una técnica denominada *truco del kernel* para construir una frontera de decisión no lineal en el espacio vectorial original mediante el mapeo de las observaciones del conjunto de entrenamiento a un espacio de producto escalar en el cual las clases son linealmente separables (Aggarwal et al. 2012).

La máquina de vectores soporte es un método de aprendizaje más flexible que el clasificador de vectores soporte. Nuevamente, debido al espacio de alta dimensionalidad en el cual se encuentra el texto, por lo general en la CT no se suele aprender una frontera de decisión no lineal, ya que un algoritmo de aprendizaje con la posibilidad de aprender un margen blando, como lo hace el clasificador de vectores soporte, suele generalizar bien (Pilászy 2005) (Hastie et al. 2009). Es decir, en el problema de la CT, la complejidad adicional de la clasificación no lineal no tiende a brindar mejores resultados que la clasificación lineal: se considera que los modelos de clasificación lineal son lo suficientemente expresivos para emplear en la CT (Aggarwal et al. 2012) (Yuan et al. 2012).

A lo largo del presente trabajo se notará $linSVM$[6] al algoritmo de aprendizaje del clasificador de vectores soporte.

3.2.2. Clasificación multi-valor

Para simplificar la presentación de la clasificación lineal sólo se ha considerado la clasificación binaria. A continuación, se presenta cómo extender los clasificadores binarios cuando $|\mathbb{C}| = J > 2$ y las clases no son mutuamente excluyentes. Presentar este escenario es realmente importante en la CT dado que, por lo general, un documento puede ser relevante a muchos tópicos de forma simultánea. En la clasificación multi-valor un documento puede pertenecer a varias clases simultáneamente, a una sola clase, o a ninguna de ellas. Aquí, como se explica en (Manning et al. 2008), se aprenden J clasificadores

[6]En la literatura, la máquina de vectores soporte con kernel lineal es equivalente al clasificador de vectores soporte

binarios γ_j distintos, donde cada uno de estos retorna c_j o $\bar{c_j}$. El procedimiento es como sigue:

1. Se aprende un clasificador por cada clase de $\mathbb{C}$, donde el conjunto de observaciones de entrenamiento consiste en el conjunto de documentos de la clase (etiquetas positivas) y su complemento (es decir, las etiquetas negativas).

2. Dado un documento de prueba, se aplica de forma separada cada clasificador binario aprendido en el paso anterior. La decisión de uno de los clasificadores no tiene influencia con las decisiones de los demás clasificadores.

La clasificación de texto multi-valor se considera la más empleada en la recuperación de información y se utilizará, más adelante, en el sistema propuesto.

3.2.3. Aprendizaje y predicción en la práctica

Para culminar con la CT en el MEV, se presenta el flujo de los procedimientos que se emplearán en este trabajo para (1) dada una colección de documentos junto con sus respectivas etiquetas aprender un clasificador binario mediante un algoritmo de aprendizaje supervisado y (2) dado un conjunto de documentos de prueba, predecir sus etiquetas utilizando el modelo aprendido.

Sean $\mathscr{C} = \{d_i\}_{i=1}^{n}$ una colección de documentos, $\mathbb{C}$ un conjunto de dos clases, $\mathscr{L} = \{l_{d_i}\}_{i=1}^{n}$ el conjunto de las respectivas etiquetas de los documentos de $\mathscr{C}$, donde $l_{d_i} \in \mathbb{C}$ para todo $i \in \{1, \ldots, n\}$, Γ un algoritmo de aprendizaje supervisado y $\mathscr{V} = \{g_i\}_{i=1}^{m}$ un conjunto de documentos de prueba. Luego, se realizan de forma ordenada los procedimientos de aprendizaje y predicción, los cuales se detallan en las Figuras 3.4 y 3.5, respectivamente. En dichas figuras, γ es un clasificador binario de texto, *preproc*() es la aplicación del preprocesamiento de texto detallada en la Sección 2.1.2 y $\vec{\delta}_{\mathscr{C}}$ es el vector de pesos globales aprendido mediante el análisis de la colección de documentos de entrenamiento $\mathscr{C}$. Se asume, a modo de simplificación, que la etapa *transformar* del MEV es además la encargada de la normalización de los vectores representación a sus vectores unidad con la finalidad de la compensación de longitudes.

Particularmente, como nota final, se destaca que las descritas etapas de aprendizaje y predicción se emplean en cada iteración de aprendizaje y validación[7] de un clasificador de texto con el propósito de prevenir la fuga de datos. Como se explica en (Kaufman et al. 2012), la fuga de datos, en el aprendizaje

[7]Se considera necesario resaltar que, durante el presente trabajo, la metodología propuesta es particularmente empleada al momento de aplicar k-fold CV

automatizado, se refiere cuando se utiliza información externa al conjunto de datos de entrenamiento para aprender un modelo.

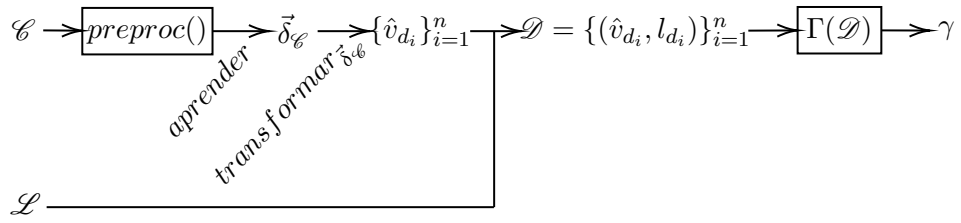

Figura 3.4: Etapa de aprendizaje de un modelo clasificador de texto, γ, aplicando el algoritmo de aprendizaje supervisado Γ en una colección de documentos etiquetados de entrenamiento. *aprender* y $transformar_{\vec{\delta}_{\mathscr{C}}}$ son etapas del MEV descritas en la Sección 2.1.1.

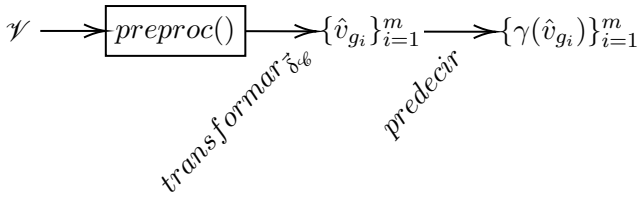

Figura 3.5: Etapa de predicción de etiquetas de un conjunto de prueba $\mathscr{V}$. γ es el clasificador de texto aprendido en el procedimiento inmediato anterior ilustrado en la Figura 3.4. $\vec{\delta}_{\mathscr{C}}$, el vector de pesos globales de $\mathscr{C}$, se utiliza para transformar todo documento de $\mathscr{V}$ a su representación vectorial.

Capítulo 4

Sistema propuesto

¿Qué sería la vida si no tuviéramos el valor de intentar algo?

— Vincent van Gogh

En este capítulo se presenta la propuesta de un asistente artificial a la matricería legal (ML). La propuesta consiste en un sistema de información de texto implementado mediante el modelo Espacio-Vectorial: principalmente, consiste en un sistema recomendador de documentos de texto que utiliza nociones de los sistemas de recomendación basados en contenido y basados en conocimiento.

Según (Russell et al. 2016), los agentes artificiales son aquellos que intentan imitar tareas cognitivas que los seres humanos realizan como por ejemplo, aprender y resolver problemas. Con este propósito se modela el sistema que se propone a lo largo del presente capítulo. Dada una empresa e, este agente se ejecuta a diario con la finalidad de poblar la matriz legal de dicha empresa imitando los pasos que se describieron en la Sección 1.1. Particularmente, el mencionado agente recupera normativas publicadas en boletines oficiales del gobierno y las clasifica en distintas ramas del Derecho de interés para e con el objeto de detectar aquellas que potencialmente traten acerca de actividades relevantes para la empresa en cuestión.

Como se ha detallado anteriormente, la ML es realizada por un conjunto de expertos en diversas temáticas. El objetivo principal del sistema que se propone es alivianar esta tarea reduciendo el conjunto de expertos que actualmente la ejecutan a sólo un encargado. Este último es quien evalúa las normativas sugeridas por el sistema y posee la decisión final acerca de agregar tales documentos a la matriz legal de la empresa.

Con el propósito de presentar la propuesta del sistema asistente a la ML, a continuación se propone una arquitectura conceptual de un asistente de soporte artificial en línea denominado *Sistema de Soporte a la Ingeniería Legal* (SiSIL) y se ilustra en la Figura 4.1. Este analiza periódicamente documentos

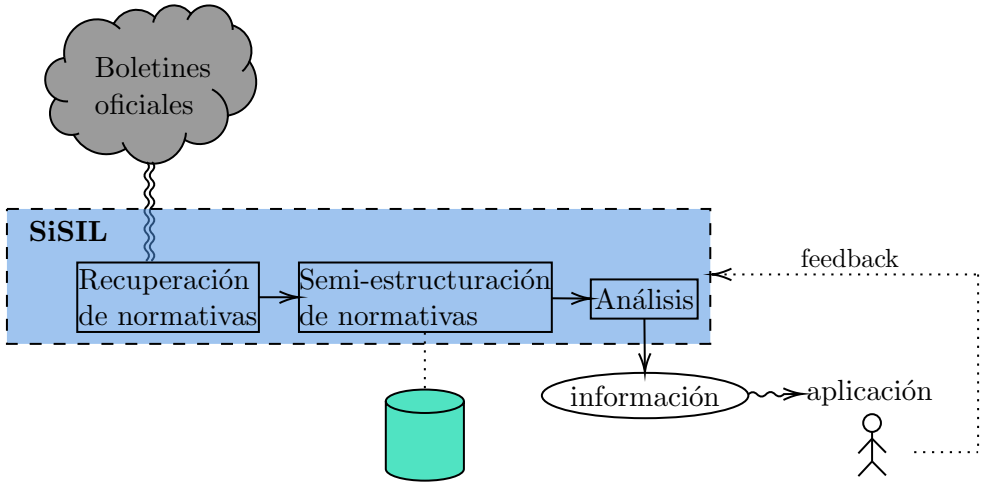

Figura 4.1: Arquitectura conceptual de un asistente, en línea, a la ingeniería legal.

normativos de numerosos boletines oficiales del gobierno con el objetivo de obtener cierta información y, de esta manera, asistir al humano experto en realizar determinadas actividades relacionadas a la ingeniería legal. SiSIL se compone de los siguientes tres módulos:

1. **Recuperación de normativas**. Los documentos normativos, como se ha detallado anteriormente, son publicados por diversos boletines oficiales del gobierno, dependiendo la jurisdicción. Cada uno de estos boletines oficiales posee su correspondiente portal web. Estos portales publican toda normativa en formato HTML[1] pero, a su vez, no comparten una formalidad a la hora de estructurar cada una de ellas; es decir, cada boletín emplea su propia forma de publicar normativas utilizando el mencionado lenguaje de marcado. No existe hasta el momento un protocolo común entre todos los boletines oficiales del gobierno sobre cómo publicar electrónicamente documentos normativos. Además, estos portales no poseen interfaces de programación de aplicaciones; por lo tanto, la recuperación debe ser realizada mediante la ejecución diaria de distintos web crawlers. Estos crawlers se deben construir específicamente para cada fuente de información. Se puede notar que algunos de estos portales emplean una estructuración HTML más fina y mejor lograda, mientras que otros implementan una estructuración considerablemente menos robusta logrando que la extracción de información mediante web crawlers sea una tarea compleja la cual involucra el empleo de técnicas de lenguajes formales.

[1] *Hypertext markup language*

2. **Semi-estructuración de normativas**. Los documentos normativos recuperados en el paso anterior son luego semi-estructurados utilizando los metadatos parseados por los distintos crawlers. Los metadatos de importancia suelen ser el tipo de normativa (ley, decreto, etc.), el número, el título, el organismo emisor, fecha de sanción o de emisión, fecha de promulgación, fecha de publicación, texto completo y anexos. Esta información, por lo general, no se encuentra etiquetada específicamente en el código HTML de las normativas, por lo tanto se deben definir expresiones regulares con el fin de encontrar patrones en cadenas de caracteres. Las normativas semi-estructuradas son inmediatamente almacenadas en una base de datos.

3. **Análisis**. Aquí, dependiendo la problemática, se aplican diversos métodos de extracción de conocimiento y minado de datos a las normativas semi-estructuradas del paso anterior y se obtiene cierta información. Utilizando esta información disponible, desarrolladores pueden implementar aplicaciones de usuario específicas de soporte a la decisión legal. Este módulo es considerado el núcleo de SiSIL, dado que emplea la inteligencia necesaria para descubrir patrones que serán luego explotados en diversas aplicaciones.

El usuario final puede brindar su crítica o feedback acerca de la calidad de los resultados retornados por la aplicación final. El feedback, sea explícito o implícito, se utiliza para que el asistente obtenga una mejor comprensión de la necesidad de información de la problemática y, así, pueda refinar la información retornada. Por ejemplo, reentrenando modelos de aprendizaje automatizado utilizados en el módulo de análisis. Este concepto propuesto de recuperación y análisis deja abierta la posibilidad de incluir distintas formas de análisis de documentos normativos de mayor complejidad, así también como nuevas aplicaciones de usuario con la finalidad de asistir a la toma de decisión legal.

En esta tesina se propone utilizar SiSIL para implementar un asistente a la ML. Por lo tanto, en la Sección 4.1, se propone un diseño específico del módulo *Análisis* con el fin de satisfacer parcialmente la necesidad de información en la ML y apoyar a expertos industriales en la mencionada actividad. Esta propuesta se ilustra en la Figura 4.2.

No todas las normativas publicadas por los diversos boletines oficiales son relevantes para toda empresa. El contexto de la empresa cumple un rol esencial en la selección de documentos normativos: se debe poseer un basto conocimiento acerca de las actividades usualmente ejecutadas por la empresa con el propósito de una recuperación de información más refinada y específica. Con la intención de afrontar la anterior problemática, en este trabajo se propone la construcción de un perfil de empresa. Sea e una empresa, el mencionado perfil se nota Ω_e y la configuración del mismo queda a cargo de un experto en las

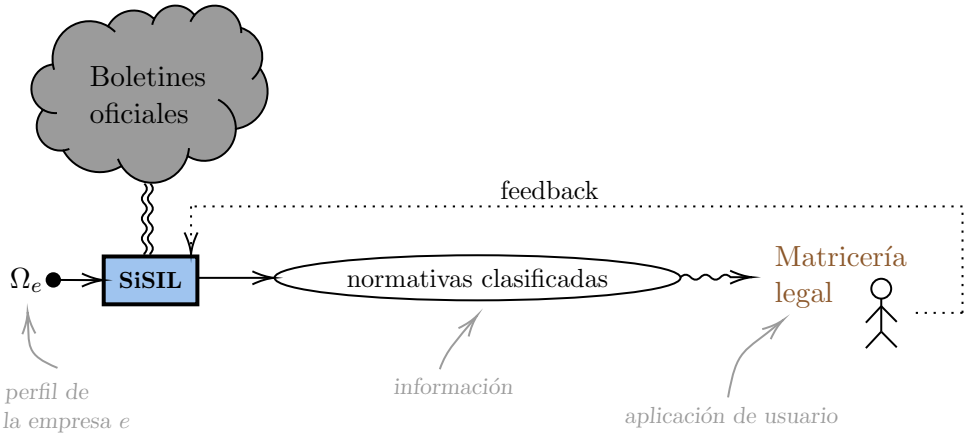

Figura 4.2: Propuesta de asistente a la ML con respecto a una empresa e.

actividades ejecutadas por e. Particularmente, Ω_e es utilizado por SiSIL con el objeto comprender, de manera parcial, el contexto de e y utilizarlo durante los procedimientos de recuperación de información y análisis de normativas.

La actividad de la ML involucra ciertas ramas del Derecho de interés para la empresa en el proceso de selección y evaluación de documentos normativos. La empresa suele interesarse por ciertas ramas por la razón de que las actividades comúnmente ejecutadas por ésta se encuentran inherentemente involucradas en el dominio de dichas ramas. Ahora bien, la construcción del mencionado perfil es una tarea designada a un usuario experto en el contexto de la empresa en cuestión. Esta tarea puede ser dificultosa de llevar a cabo ya que entender y modelar el contexto de una empresa es considerada una tarea compleja. Es decir, en escenarios tan complejos y diversos como los contextos de empresas de diferentes índoles, se considera trabajoso para los usuarios enunciar sus requerimientos. Para solventar lo anterior, en este trabajo se propone aplicar nociones de los sistemas recomendadores basados en conocimiento; es decir, emplear un diálogo entre el sistema propuesto y el usuario experto con la finalidad de recopilar requisitos y construir un perfil que represente lo más adecuadamente posible el contexto de la empresa. El diálogo anterior se divide en dos etapas, las cuales suceden en orden: (1) la selección de ramas del Derecho de principal interés para la empresa y (2) la exploración del dominio de cada rama escogida. Particularmente, durante las anteriores dos etapas, el sistema que se propone en esta tesina hace uso de una fuente de conocimiento externa: el Tesauro del Derecho Argentino[2] (TDA), desarrollado por el Sistema Argentino de Información Jurídica[3] (SAIJ). El TDA es

[2] `http://admin.tcda.infojus.gov.ar/saij/sobre.php`

[3] Base de datos de documentación jurídica dependiente de la Secretaría de Justicia del Ministerio de Justicia y Derechos Humanos de la Nación. Su portal web se encuentra en `http://www.saij.gob.ar`

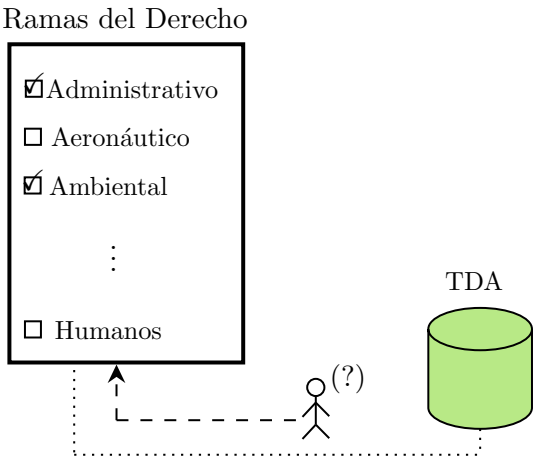

Figura 4.3: Primera etapa del diálogo entre el usuario experto y el sistema asistente.

un vocabulario controlado el cual comprende 16 ramas del Derecho. Más precisamente, el TDA es un sistema de organización del conocimiento de ramas del Derecho y tiene como propósito asistir la búsqueda temática de la base de datos de documentos legales del SAIJ. Un sistema de organización de conocimiento es un conjunto de elementos, usualmente estructurados y controlados, utilizado para describir objetos y explorar colecciones. Los tesauros, principalmente, se utilizan para implementar interfaces de búsqueda más inteligentes y organizar el conocimiento de cierto dominio (Yu 2011). Especialmente, el TDA está implementado mediante el estándar SKOS[4] de la Web Semántica[5].

A continuación, se describe el mencionado diálogo entre el sistema asistente y un usuario u experto en el contexto de cierta empresa e con el fin de construir el perfil Ω_e. En la primer etapa del dialogo, el sistema presenta a u las 16 ramas del Derecho disponibles en base a la organización de la doctrina que se especifica en el TDA. De las anteriores ramas, u selecciona aquellas principalmente relevantes acorde a las actividades que e suele ejecutar. En la Figura 4.3 se ilustra un ejemplo de la mencionada primera etapa del diálogo. Esta información brindada por el usuario se utiliza con el fin de seleccionar, primeramente, aquellas normativas consideradas pertenecientes a alguna o algunas de dichas ramas, dentro de una cadena de normativas recuperadas de distintos boletines oficiales. Ahora bien, es importante recalcar que no todas las normativas categorizadas mediante alguna rama del Derecho de interés son del todo relevantes para e. La ML aplica un criterio más selectivo a la hora de decidir si un documento normativo es exigible a una empresa: para esto, los encargados de realizar la mencionada actividad se sumergen en el

[4]Del inglés *Simple Knowledge Organization System*, SKOS es un vocabulario para representar sistemas de organización de conocimiento para su publicación en la web (Yu 2011)

[5]`http://www.w3.org/standards/semanticweb/`

contexto en el cual e se involucra con la finalidad de refinar su criterio de selección de normativas. Como se ha comentado anteriormente, las actividades que realiza la empresa en el día a día son las que definen el contexto de la misma. Además, como se notó al comienzo, estas actividades se encuentran inherentemente involucradas en el dominio de cada rama del Derecho. Por lo tanto, en esta tesina se propone refinar la selección de normativas mediante una exploración algo más exhaustiva con respecto a cada rama del Derecho escogida por u en la etapa anterior. Para llevar a cabo la respectiva tarea, el sistema utiliza, nuevamente, el TDA como base de conocimiento. El mencionado tesauro organiza el conocimiento del dominio de cada rama del Derecho empleando una jerarquía de términos específicos[6]. Esta segunda etapa del diálogo entre u y el sistema se considera similar a la primera, sólo que ahora se presentan distintos términos específicos para cada rama escogida por u durante la mencionada primer etapa. Un ejemplo de la recientemente detallada segunda etapa del diálogo, en donde se considera sólo dos ramas del Derecho, se ilustra en la Figura 4.4. En este caso, se puede observar la asistencia del sistema en la exploración de terminología específica del Derecho Laboral y el Derecho Civil. Esta metodología propuesta de exploración del dominio de cada rama del Derecho de interés, tiene como objetivo refinar el proceso de selección de normativas mediante la detección de tópicos potencialmente relevantes con respecto a ciertas actividades de e. Para ejemplificar lo anterior, si e es una metalúrgica, sus empleados se ven eventualmente implicados en diversos accidentes ocasionados por distintas maquinarias dentro de la empresa. Por lo tanto, e está sumamente interesada en detectar normativas que traten acerca del tópico "accidentes de trabajo", término específico del dominio del Derecho Laboral. En cambio, se puede considerar que el anterior tópico no es sumamente relevante en empresas de, por ejemplo, índole informática.

Sea $\mathbb{D}$ el conjunto de las 16 ramas del Derecho consideradas por el TDA, el perfil Ω_e se especifica como la tupla

$$\Omega_e = (R, M, B),$$

donde:

- R es un conjunto de pares (r, T), en donde $r \in \mathbb{D}$ es una rama de interés seleccionada por u junto con su respectivo conjunto T de términos más específicos escogidos durante el diálogo de solicitud de requerimientos. Particularmente, en este trabajo se utilizan las siguientes dos notaciones: $R^\star = \{r \in \mathbb{D} \mid (r, T) \in R,\ p.a.\ T\}$ y T^r como el conjunto T tal que $(r, T) \in R$.

[6]Por ejemplo, "ley procesal" es uno de los términos más específicos del Derecho Procesal según el TDA

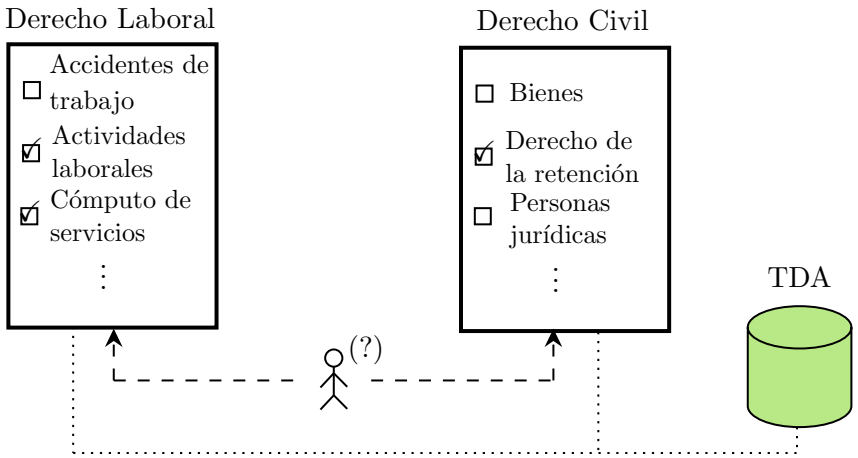

Figura 4.4: Ejemplo de la segunda etapa del diálogo entre el usuario experto y el sistema asistente. El usuario experto optó por explorar dos ramas del Derecho.

- $M = \bigcup_{r \in R^\star} \{M_r\}$ es el repositorio inicial[7] de normativas relevantes para e; es decir, M es la matriz legal de e, donde $d \in M_j$ si d es categorizada como del Derecho j.
- B es el conjunto de boletines oficiales considerados por e.

En las Secciones 4.1 y 4.2 se describe la tarea que realiza el módulo *Análisis* de SiSIL y la aplicación de usuario final, respectivamente, con el propósito de semi-automatizar la actividad de la matricería legal.

4.1. Clasificación artificial de normativas

Sea e una empresa; luego, la necesidad de información en la ML es estática y se puede describir como sigue:

> "Obtener, de determinados boletines oficiales, documentos normativos potencialmente relevantes a las actividades propias e inherentes a la actividad productiva de e".

Como se detalló anteriormente, en este trabajo se utiliza SiSIL para modelar la propuesta de un agente asistente a la ML; por lo tanto, a lo largo de la presente sección se propone un diseño específico del módulo *Análisis* con la finalidad de afrontar la anterior necesidad de información. Este módulo es quien evalúa cada normativa recuperada de los boletines oficiales configurados en Ω_e (precisamente en el conjunto B) con el fin de predecir la relevancia con

[7] Documentos normativos relevantes para la empresa hasta la fecha

respecto al contexto de e. Particularmente, en esta tesina se propone que dicho módulo sea el responsable de (1) la clasificación de normativas en ramas del Derecho de interés para e y (2) la detección de tópicos potencialmente relevantes al contexto de e con respecto al dominio de dichas ramas.

4.1.1. Clasificación en ramas del Derecho

Las ramas del Derecho, conjunto $\mathbb{D}$, no son mutuamente excluyentes[8]. Luego, la clasificación de documentos normativos en ramas del Derecho es un problema de clasificación multi-valor: se deben aprender $|\mathbb{D}|$ clasificadores binarios; es decir, un clasificador binario de texto γ_r para cada $r \in \mathbb{D}$. La metodología para aprender cada uno de estos clasificadores es la misma y se propone como sigue.

Como primer medida, se recuperan normativas para cada clase de $\mathbb{D}$. A lo largo de este trabajo, al conjunto anterior se lo nota Λ. Es decir, Λ es un conjunto de pares (d, e) donde d es un documento normativo y $e \in \mathbb{D}$ es una rama del Derecho a la cual d pertenece. Además, se supondrá que el anterior conjunto es una partición: $\bigcup_{r\in\mathbb{D}} \Lambda_r = \Lambda$ y $\bigcap_{r\in\mathbb{D}} \Lambda_r = \emptyset$, donde $\Lambda_r = \{d \mid (d, r) \in \Lambda\}$.

Ahora bien, sea $r \in \mathbb{D}$, con el fin de aprender un clasificador de texto γ_r tal que $\gamma_r(d) \in \{r, \bar{r}\}$, se procede con la construcción de un conjunto de observaciones positivas (las normativas categorizadas mediante la rama r) y observaciones negativas (muestras aleatorias de documentos de los restantes $|\mathbb{D}| - 1$ conjuntos de normativas Λ_i, con $i \neq r$). Luego, se define un conjunto de observaciones $\mathscr{D}_{\Lambda_r} = \mathscr{D}^+_{\Lambda_r} \cup \mathscr{D}^-_{\Lambda_r}$, donde:

$$\mathscr{D}^+_{\Lambda_r} = \{(d, r) \mid d \in \Lambda_r\}$$
$$\mathscr{D}^-_{\Lambda_r} = \{(d, \bar{r}) \mid d \in \bigcup_{i\in\mathbb{D}-\{r\}} rand(\left\lfloor \frac{|\Lambda_r|}{|\mathbb{D}| - 1} \right\rfloor, \Lambda_i)\},$$

y *rand* se define como

$$rand(n, C) = \begin{cases} S \subset C \text{ aleatorio tal que } |S| = n & \text{si } n < |C|, \\ C & \text{en otro caso.} \end{cases}$$

Esta propuesta de construcción del conjunto $\mathscr{D}_{\Lambda_r}$ tal que posea todas las observaciones positivas y un subconjunto aleatorio de observaciones negativas tiene como propósito la construcción de un conjunto aproximadamente balanceado, en un escenario de clases desbalanceadas. La Figura 4.5 ilustra la mencionada propuesta. La estrategia anterior se puede considerar derivada de la técnica denominada submuestreo aleatorio[9].

[8] Por ejemplo, el decreto nacional 720/2014 es considerado por el SAIJ como perteneciente al Derecho Ambiental y al Derecho Civil

[9] La técnica de submuestreo aleatorio equilibra las distribuciones de clase descartando, al azar, instancias de la clase mayoritaria (Borovicka et al. 2012)

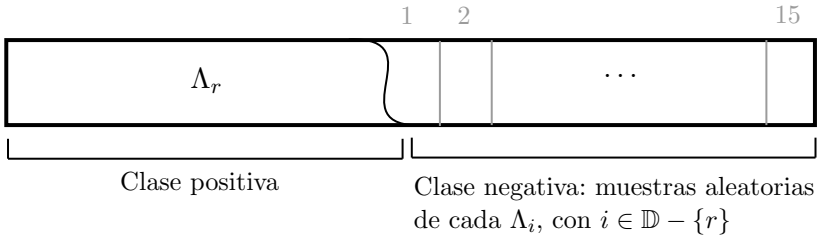

Figura 4.5: Construcción del conjunto aproximadamente balanceado de observaciones $\mathscr{D}_{\Lambda_r}$.

Una vez construido $\mathscr{D}_{\Lambda_r}$, se procede con el aprendizaje y validación de γ_r utilizando las metodologías descritas en la Sección 3.2.3, aplicando $linSVM$ como algoritmo de aprendizaje. Particularmente, se propone realizar los siguientes pasos:

1. **Partición del conjunto total**. Se procede con la partición de $\mathscr{D}_{\Lambda_r}$, de forma aleatoria y estratificada[10], en conjuntos disjuntos de entrenamiento y validación, los cuales se notan $\mathscr{T}$ y $\mathscr{V}$, respectivamente. $\mathscr{V}$ es totalmente aislado de proceso de aprendizaje y se utiliza para estimar el error de generalización del modelo aprendido.

2. **Ajuste del hiperparámetro de regularización**. Como se detalló en la Sección 3.2.1, $linSVM$ posee un hiperparámetro de regularización denominado C. Sea V_C un conjunto de posibles configuraciones de C. Utilizando cada configuración de V_C, se aplica k-fold CV estratificado[11] en el conjunto de entrenamiento $\mathscr{T}$ resultando en múltiples modelos y estimaciones de errores de generalización. Este procedimiento, usualmente, se denomina búsqueda en cuadrícula.

3. **Entrenamiento**. En este paso, como primer medida se toma la configuración del hiperparámetro C con mejor resultado de generalización del paso anterior. Luego, configurando C con el valor anterior, se aplica el algoritmo de aprendizaje $linSVM$ en el conjunto de observaciones de entrenamiento $\mathscr{T}$ para obtener el clasificador γ_r; es decir, $\gamma_r = linSVM(\mathscr{T})$.

4. **Validación**. Utilizando el conjunto de observaciones $\mathscr{V}$ apartado al comienzo, se evalúa la generalización del modelo aprendido, γ_r, utilizando métricas presentadas en la Sección 2.2.

[10]En un problema de clasificación, el término *estratificación* se referiere a que ambos conjuntos de entrenamiento y prueba contengan aproximadamente el mismo porcentaje de muestras de cada clase que el conjunto completo (Alpaydin 2010)

[11]k-fold CV estratificado es una variación del método k-fold CV en la cual cada fold contiene aproximadamente el mismo porcentaje de muestras de cada clase

Esta propuesta de aprendizaje de clasificadores binarios en ramas del Derecho se utiliza para aprender todo γ_r tal que $\gamma_j(d) \in \{j, \bar{j}\}$ con $j \in \mathbb{D}$.

4.1.2. Detección de tópicos relevantes

Una vez clasificada cada normativa recuperada de los boletines oficiales en ramas del Derecho de interés para la empresa, en el siguiente paso se continúa analizando cada una de las normativas con el objetivo de encontrar patrones que indiquen potencial relevancia con respecto a las actividades que la empresa ejecuta para llevar a cabo, de esta forma, una selección de normativas más refinada. Para afrontar esta problemática, como se comentó al comienzo del presente capítulo, en esta tesina se propone que el usuario experto en el contexto de la empresa explore el dominio de cada rama del Derecho de interés mediante la ayuda del TDA como base de conocimiento. De esta manera, se considera que el sistema obtendrá más información acerca de las actividades que la empresa realiza y seleccionará aquellas normativas las cuales potencialmente traten sobre términos más específicos de cada rama del Derecho escogidos por el usuario experto.

Partiendo del perfil Ω_e construido para la empresa e, para cada $r \in R^\star$, se aprende un clasificador de texto σ_t^r, tal que $\sigma_t^r(d) \in \{t, \bar{t}\}$, para cada $t \in T^r$. Es decir, por cada rama $r \in R^\star$, se aprende un clasificador por cada término más específico del dominio de r escogido por el usuario en el diálogo de solicitud de requerimientos. Cada uno de los anteriores modelos se aprende de manera análoga construyendo un conjunto de observaciones $\mathscr{D}_t$ utilizando sólo los documentos del conjunto Λ_r, ya que se busca aprender sobre un término específico del dominio del Derecho r. Por lo tanto, se debe enfrentar nuevamente un problema de clases desbalanceadas: la cardinalidad del conjunto de todos los $d \in \Lambda_r$ tales que además están etiquetados mediante la clase t (observaciones positivas), notado Λ_r^t, será menor que la cardinalidad del conjunto de los restantes documentos normativos del conjunto Λ_r (observaciones negativas). La Figura 4.6 intenta ilustrar la anterior problemática. Con lo cual, se propone aplicar submuestreo aleatorio a la clase mayoritaria; es decir, a la clase negativa. De esta forma, se construye $\mathscr{D}_t = \mathscr{D}_t^+ \cup \mathscr{D}_t^-$, donde:

$$\mathscr{D}_t^+ = \{(d, t) \mid d \in \Lambda_r^t\}$$
$$\mathscr{D}_t^- = \{(d, \bar{t}) \mid d \in rand(|\Lambda_r^t|, \Lambda_r - \Lambda_r^t)\},$$

Se puede notar que $\mathscr{D}_t \subset \Lambda_r$.

Una vez construido $\mathscr{D}_t$ se procede con el aprendizaje del clasificador binario de texto σ_t^r para luego validarlo tal como se explicó en la sección anterior, utilizando nuevamente $linSVM$ como algoritmo de aprendizaje.

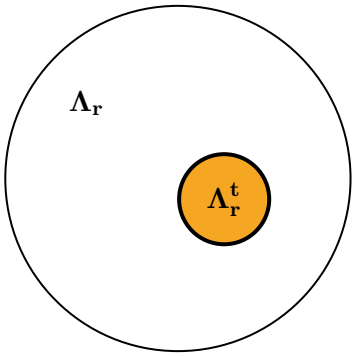

Figura 4.6: Ejemplo ilustrativo del conjunto Λ_r^t. Este conjunto contiene todas las normativas de Λ_r etiquetadas mediante el término más específico t del dominio del Derecho r.

4.1.3. Implementación

Hasta aquí, se han descrito las tareas asignadas al módulo *Análisis* de SiSIL para afrontar la necesidad de información en la ML. Estos procedimientos analíticos se aplican a cada normativa recuperada de los configurados boletines oficiales. El pseudocódigo del Algoritmo 1 propone lo anterior. La Figura 4.7 es un diagrama meramente ilustrativo del mencionado algoritmo, donde $\{r_1, \ldots, r_n\} \subseteq R^\star$ y la notación t^r indica que el término específico t pertenece al conjunto T^r, con $r \in R^\star$. En la mencionada figura se puede observar cómo se procede con una clasificación más exhaustiva de la normativa d a partir de que esta última es, primeramente, clasificada mediante alguna rama del Derecho de interés para la empresa. Esta clasificación tiene como propósito una selección más refinada de documentos normativos para afrontar la problemática de la ML: una normativa se considera potencialmente relevante si además de ser categorizada por una rama $r \in R^\star$ se considera que trata sobre algún tópico más específico del dominio de r.

A continuación, se hace foco en el diseño e implementación de la aplicación de usuario; es decir, la interacción entre la información brindada por la presente etapa de análisis y el usuario experto.

4.2. La matricería legal como aplicación de usuario

La información brindada por el módulo *Análisis* de SiSIL acerca de cada normativa recuperada de los distintos boletines oficiales es utilizada en la sugerencia de documentos; es decir, en la aplicación final de usuario. El pseudocódigo de la aplicación, la cual se ejecuta diariamente e involucra a un experto en el contexto de la empresa, se propone en el Algoritmo 2, en donde el procedimiento *análisis*() hace referencia al Algoritmo 1. Un ejemplo ilustrativo del mencionado algoritmo se presenta en la Figura 4.8. En dicha

Algoritmo 1: Análisis de documento normativo

Datos: d, documento normativo en texto plano; Ω_e, perfil de la empresa e; $\{\gamma_r\}_{r \in R^\star}$ y $\{\sigma_t^r\}_{r \in R^\star, t \in T^r}$, clasificadores aprendidos offline

Resultado: I, conjunto de pares $(r, T) \in R^\star \times T^r$ donde r indica que d fue categorizado como perteneciente a la rama r del Derecho y $T \neq \emptyset$ contiene los términos más específicos del dominio de r acerca de los cuales el documento d potencialmente trata

$I \leftarrow \emptyset$
`// Clasificación multi-valor de normativa en ramas del Derecho de interés`
$M \leftarrow \{r \in R^\star \mid \gamma_r(d) = r\}$
para cada $r \in M$ **hacer**
  `// Detección de tópicos específicos con respecto a la rama` r `del Derecho`
  $M' \leftarrow \{t \in T^r \mid \sigma_t^r(d) = t\}$
  `// Si la normativa no trata sobre algún tópico más especifico, se decide descartarla`
  si $M' \neq \emptyset$ **entonces**
    $I \leftarrow I \cup \{(r, M')\}$
  fin
fin
devolver I

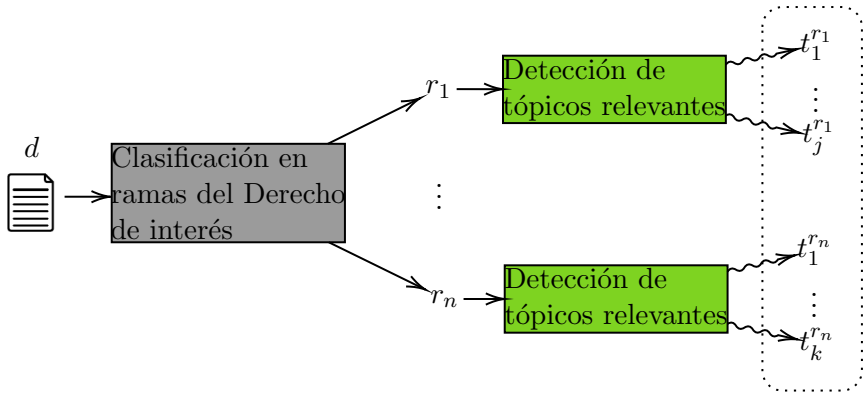

Figura 4.7: Propuesta de proceso de análisis de una normativa d. Las hojas del árbol son la respuesta de clasificación de d con respecto al dominio de cada rama del Derecho de interés para la empresa.

Algoritmo 2: Matricería legal

Datos: $\Omega_e = (R, M, B)$, perfil de e construido por usuario experto

```
// Recuperación de normativas de boletines configurados
S ← crawl(B)
para cada d ∈ S hacer
    // Análisis de la normativa con respecto a los
       requerimientos de la empresa
    I ← análisis(d, Ω_e)
    si I ≠ ∅ entonces
        // El sistema considera que d es potencialmente
           relevante y, por lo tanto, se presenta al usuario
           junto a la respectiva información obtenida
        presentarNormativa(d, I)
        si usuarioSelecciona(d) entonces
            // El usuario considera que d es realmente
               relevante. Luego, el documento se agrega al
               repositorio junto con sus etiquetas
            actualizarMatrizLegal(M, (d, I))
        fin
    fin
fin
```

figura se supone que la empresa e está interesada en el Derecho Ambiental y el Derecho Laboral y, en especial, en ciertos términos (tópicos) específicos, respectivamente. El usuario interactúa constantemente con los documentos sugeridos por el sistema y selecciona aquellos que a su criterio son relevantes para el contexto de e.

El pseudocódigo de la aplicación de la ML, Algoritmo 2, se puede resumir en lenguaje natural como sigue. Sean e un empresa, u el usuario encargado en utilizar el sistema, $\Omega_e = (R, M, B)$ el perfil de e y S una cadena de normativas recuperadas luego, para cada $d \in S$:

1. El sistema evalúa d, como se detalló en el Algoritmo 1, con el objeto de detectar si el documento se considera potencialmente perteneciente a alguna o algunas de las ramas del Derecho de interés para e y, en el caso que lo haga, detectar qué término o términos específicos en particular. En el caso de que el análisis de d no haya detectado información relevante para el contexto de e, la normativa se descarta. En caso contrario, se procede al siguiente paso.

2. La normativa, junto con sus respectivas etiquetas, es presentada a u mediante una interfaz de usuario, tal como se ilustra en la Figura 4.8. Aquí, u evalúa la relevancia de d aplicando sus conocimientos acerca del

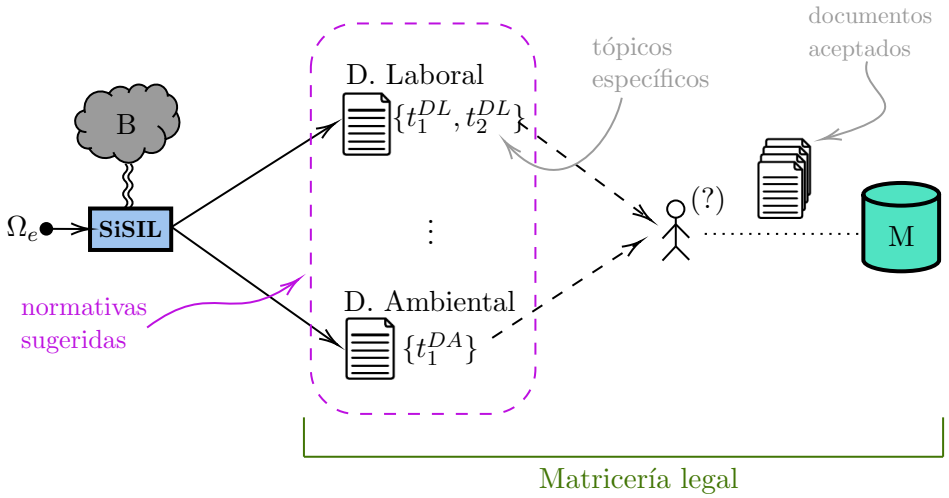

Figura 4.8: Ejemplo ilustrativo de la aplicación de la matricería legal en una empresa e utilizando SiSIL como agente de soporte artificial. e sólo está interesada en el Derecho Ambiental (DA) y el Derecho Laboral (DL) y específicamente sobre ciertos tópicos para cada rama en particular.

contexto de la empresa y considerando la información obtenida mediante las etiquetas otorgadas por el respectivo análisis artificial aplicado a la normativa. En el caso de que u no considere relevante a d, esta última se descarta. En caso contrario, el sistema se encarga de actualizar la matriz legal de e incluyendo la nueva normativa y haciendo uso de sus etiquetas. Por ejemplo, si una normativa fue etiquetada mediante una rama $r \in R^\star$ junto con dos términos más específicos $t_1, t_2 \in T^r$, luego $M_r \leftarrow M_r \cup \{(d, \{t_1, t_2\})\}$ donde $M_r \in M$.

Cabe destacar que la función *usuarioSelecciona*(), mediante el comportamiento del usuario (es decir, mediante la selección o el descarte de normativas sugeridas), es la encargada de enviar feedback a SiSIL con la finalidad de una futura mejor comprensión acerca de los requerimientos del usuario. Es decir, en este caso, las normativas que son consideradas potencialmente relevantes por SiSIL pero que luego son descartadas por el usuario, en realidad se almacenan en un repositorio distinto a la matriz legal de e para, eventualmente, refinar el análisis de futuros documentos normativos reentrenando clasificadores de texto.

Capítulo 5

Experimentación

Con el propósito de aprender los distintos clasificadores de texto descritos en el capítulo anterior, en la Figura 5.1 se presenta información sobre la recuperación de documentos normativos de la base de datos legislativa del Sistema Argentino de Información Jurídica (SAIJ) con respecto a las 16 ramas del Derecho mencionadas en el Capítulo 4. Los 16 conjuntos de normativas son disjuntos. En total, se dispone de 26520 documentos normativos para experimentación, los cuales se encuentran distribuidos aproximadamente entre los años 1900 y 2019. Dado que el SAIJ no dispone de una interfaz de programación de aplicaciones[1] con la cual realizar consultas y recuperar normativas, se optó por implementar especialmente un web crawler mediante la herramienta *Scrapy*[2].

La descripción de un ejemplo de empresa junto con los resultados de validación de distintos clasificadores se presenta a continuación.

5.1. Caso de estudio

La fase de experimentación de la presente tesina se desarrolló en conjunto con un grupo de profesionales de la Ley, los cuales configuraron un determinado perfil de empresa. Dicha empresa, la cual se notará e, es una industria aceitera localizada en la ciudad de San Lorenzo, Santa Fe, Argentina. El perfil de e, $\Omega_e = (R, M, B)$, fue configurado como sigue:

- El conjunto R está formado por los siguientes dos pares:
  - $(Derecho\ Ambiental, \{desechos\ peligrosos\})$ y
  - $(Derecho\ Laboral, \{accidentes\ de\ trabajo\})$.

  Es decir, la empresa está interesada en la recuperación de normativas pertenecientes al Derecho Ambiental y al Derecho Laboral pero, princi-

[1] Del inglés, *application programming interface* (API)

[2] Scrapy es un framework de web crawling, de código abierto, escrito en Python

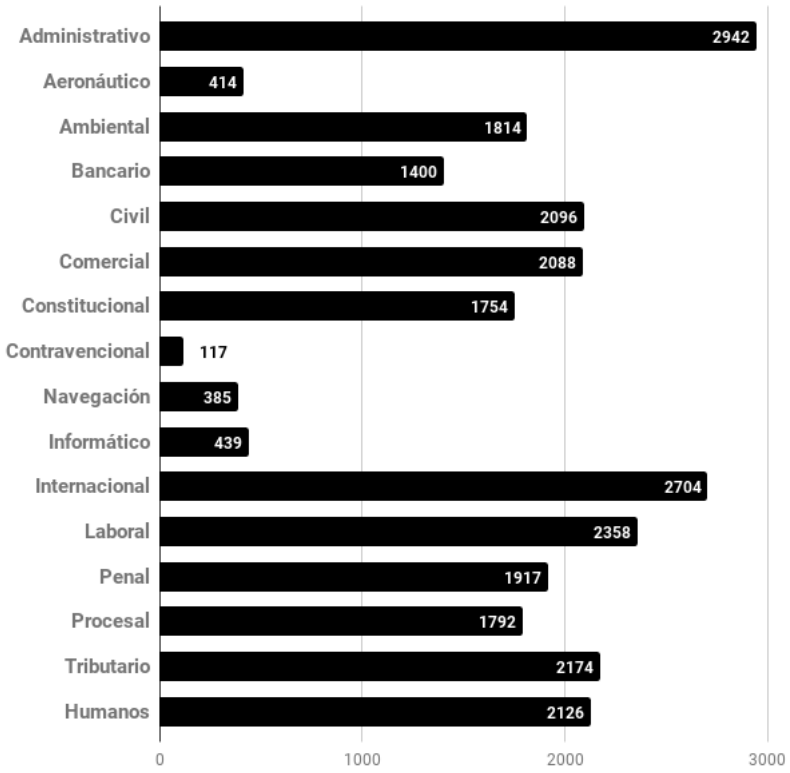

Figura 5.1: Número de normativas legislativas recuperadas para cada una de las 16 ramas del Derecho consideradas por el Tesauro del Derecho Argentino del SAIJ.

palmente, en aquellas que traten sobre desechos peligrosos y accidentes de trabajo, respectivamente.

- La matriz legal inicial de e, M, se compone por 33 y 36 documentos normativos del Derecho Ambiental y Derecho Laboral, respectivamente. Estos documentos pertenecen al repositorio de normativas, hasta la fecha, relevantes para e y fueron brindados por los mencionados expertos.

- El Boletín Oficial de la República Argentina[3], el Boletín Oficial de la Provincia de Santa Fe[4] y el Boletín Oficial de la Ciudad de San Lorenzo[5], Provincia de Santa Fe, son las tres fuentes de información configuradas en el conjunto B.

[3] `http://www.boletinoficial.gob.ar/`
[4] `http://www.santafe.gob.ar/boletinoficial/`
[5] `http://sanlorenzo.gob.ar/ordenanzas/`

Como primer medida, se implementaron tres crawlers para cada portal web de los boletines en B. Estos crawlers se implementaron, nuevamente, utilizando la herramienta Scrapy. Estos se ejecutan de forma automática para recuperar todas las normativas del día e inmediatamente almacenarlas en una base de datos[6]. Como se ha detallado en el Capítulo 4, el sistema propuesto para asistir a la ML en la empresa e utiliza el Sistema de Apoyo a la Ingeniería Legal (SiSIL) propuesto, para sugerir documentos normativos en tiempo real. SiSIL analiza periódicamente un conjunto de documentos de entrada mediante la aplicación de diversos clasificadores de texto. En este caso de ejemplo particular, debido a los requerimientos de la empresa especificados en Ω_e, se aprendieron cuatro modelos clasificadores de texto:

- Con respecto al Derecho Ambiental (DA), se aprendieron dos modelos clasificadores binarios de texto γ_{DA} y σ_{dp}^{DA}, donde el primero detecta normativas potencialmente pertenecientes al DA y el segundo detecta normativas potencialmente relevantes al tópico específico "desechos peligrosos" (dp).

- De forma análoga, con respecto al Derecho Laboral (DL), se aprendieron dos modelos clasificadores de texto γ_{DL} y σ_{at}^{DL}, donde at hace referencia al término más específico "accidentes de trabajo".

Cada uno de los anteriores modelos se aprendieron y validaron tal como se describe en la Sección 4.1. En el Cuadro 5.1 se presenta información acerca de la construcción de los conjuntos de observaciones utilizados para el aprendizaje y validación de los mencionados modelos. En la Sección 5.1.1 y 5.1.2

Cuadro 5.1: Información de los conjuntos de observaciones utilizados.

Conjunto de observaciones	# obs. positivas	# obs. negativas
Derecho Ambiental	1814	1797
Desechos peligrosos	241	241
Derecho Laboral	2358	2315
Accidentes de trabajo	228	228

se presentan los resultados de validación obtenidos de los cuatro modelos clasificadores. En cada caso, se determinó una configuración para el valor del hiperparámetro C de $linSVM$ realizando la búsqueda en cuadrícula en el conjunto de valores[7] $\{0.01, 0.1, 1, 10, 100\}$ aplicando 10-fold CV estratificado en el conjunto de observaciones designado como entrenamiento, tal como se especificó en la Sección 4.1. Gracias a la metodología que se propuso durante

[6]Particularmente, se utilizó una base de datos NoSQL

[7]En la práctica, utilizar una escala logarítmica de posibles valores se considera un método rápido para identificar buenas configuraciones para el hiperparámetro C (Alpaydin 2010)

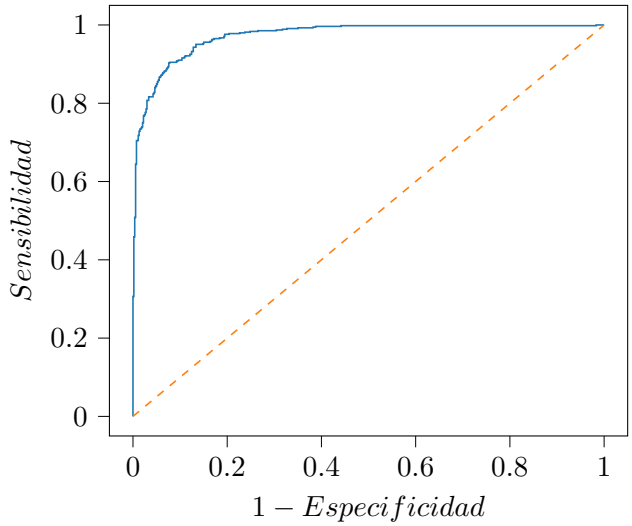

Figura 5.2: Curva ROC del modelo γ_{DA}

la mencionada sección acerca de cómo entrenar y validar cada modelo clasificador, los conjuntos de observaciones para validar cada uno de ellos resultan aproximadamente balanceados; con lo cual, se considera que el resultado de la medición de la exactitud (ACC) en el umbral de decisión por defecto de $linSVM$ (es decir, $t = 0$) es la más atractiva para resumir el rendimiento del clasificador. Se destaca que no se optó por modificar el umbral de decisión por defecto de cada modelo aprendido por la razón de que, a lo largo de la experimentación del presente trabajo, no se dispuso de forma concreta de un modelo de utilidad[8] del usuario a partir del cual modificar dicho umbral. Igualmente, a modo de resumen, para cada uno de los modelos aprendidos se presenta la curva ROC como medida gráfica de rendimiento general.

5.1.1. Derecho Ambiental

El modelo γ_{DA} se entrenó configurando el hiperparámetro $C = 1$. En la Figura 5.2 se presenta la curva ROC del modelo aprendido, evaluado utilizando $\sim 30\,\%$ de las observaciones del conjunto total. La matriz de confusión que resume el rendimiento del clasificador γ_{DA} en el umbral de decisión $t = 0$ se ilustra en la Figura 5.3. De esta matriz se estima que $ACC = 0.91$, $REC = 0.90$ y $PREC = 0.92$.

De forma similar, σ_{dp}^{DA} se entrenó configurando $C = 1$ y en la Figura 5.4 se ilustra su respectiva curva ROC del modelo evaluado con $\sim 25\,\%$ del conjunto total. En la Figura 5.5 se ilustra la matriz de confusión del modelo en el umbral

[8] Aquel que especifica los requerimientos y expectativas de un usuario en un sistema de información

		Predicho		
		$\overline{DA}$	DA	total
Real	$\overline{DA}$	494	45	539
	DA	52	493	545
	total	546	538	

Figura 5.3: Matriz de confusión de γ_{DA} en el umbral de decisión $t = 0$.

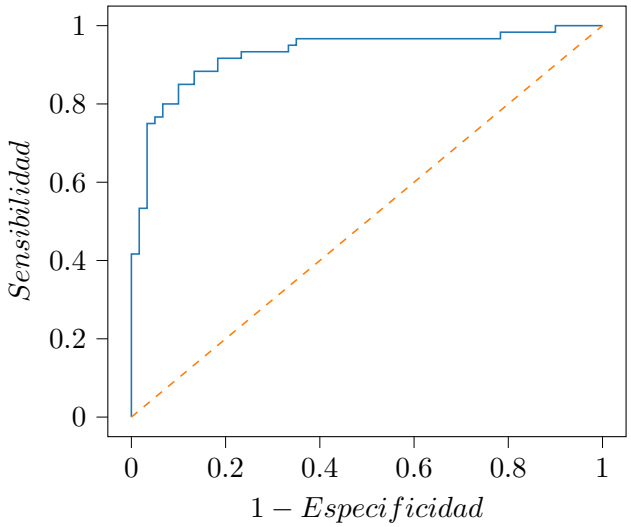

Figura 5.4: Curva ROC del modelo σ_{dp}^{DA}.

de decisión $t = 0$ y se estima que $ACC = 0.87$, $REC = 0.83$ y $PREC = 0.89$.

5.1.2. Derecho Laboral

Con respecto al dominio del Derecho Laboral, γ_{DL} se entrenó configurando el hiperparámetro $C = 1$. La Figura 5.6 ilustra la curva ROC del mencionado modelo clasificador, evaluado utilizando $\sim 30\,\%$ de las observaciones del conjunto total. La matriz de confusión que resume el clasificador en el umbral de decisión $t = 0$ se presenta en la Figura 5.7. Mediante esta matriz, se estima $ACC = 0.88$, $REC = 0.87$ y $PREC = 0.89$.

Ahora bien, el modelo σ_{at}^{DL} se entrenó configurando $C = 10$ y en la Figura 5.8 se ilustra la gráfica ROC de dicho modelo evaluado con el $25\,\%$ del conjunto

		Predicho		
		$\overline{dp}$	dp	total
Real	$\overline{dp}$	54	6	60
	dp	10	50	60
	total	64	56	

Figura 5.5: Matriz de confusión de σ_{dp}^{DA} en el umbral de decisión $t = 0$.

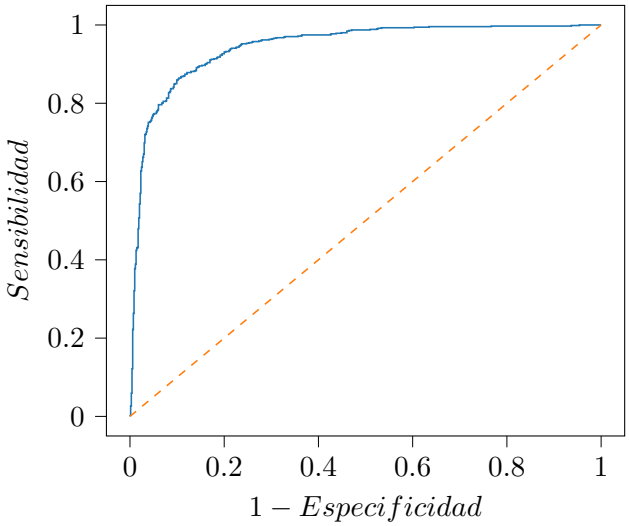

Figura 5.6: Curva ROC del modelo γ_{DL}.

		Predicho		
		$\overline{DL}$	DL	total
Real	$\overline{DL}$	619	76	695
	DL	94	613	707
	total	713	689	

Figura 5.7: Matriz de confusión de γ_{DL} en el umbral de decisión $t = 0$.

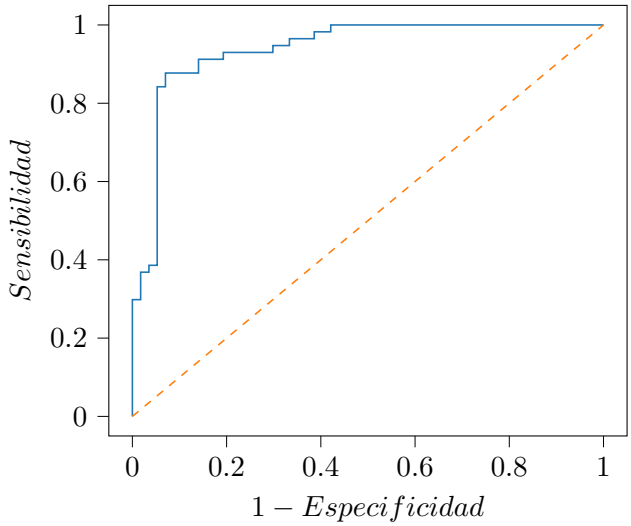

Figura 5.8: Curva ROC del modelo σ_{at}^{DL}.

		Predicho		
		$\overline{at}$	at	total
Real	$\overline{at}$	48	9	57
	at	5	52	57
	total	53	61	

Figura 5.9: Matriz de confusión de σ_{at}^{DL} en el umbral de decisión $t = 0$.

total de observaciones. Luego, en la Figura 5.9 se presenta la correspondiente matriz de confusión en el umbral de decisión $t = 0$. Mediante la información que brinda la anterior matriz, se estima que $ACC = 0.88$, $REC = 0.91$ y $PREC = 0.85$.

5.1.3. Resultados

Con el propósito de aplicar el caso de estudio a normativas no incluidas en los conjuntos de datos construidos para entrenar y validar los modelos clasificadores presentados anteriormente, se recopilaron documentos normativos de las tres fuentes de información configuradas en Ω_e. Estos documentos, no etiquetados, fueron publicados desde principios del mes de enero hasta fines de mayo de 2019. De las anteriores normativas, se decidió trabajar sobre 4766

documentos publicados por el Boletín Oficial de la República Argentina. A continuación, se presentan los resultados de la simulación offline de los documentos candidatos a poblar la matriz legal de la empresa, considerando por separado el Derecho Ambiental y el Derecho Laboral en el Cuadro 5.2 y 5.3, respectivamente. Dado que sólo se seleccionan aquellos documentos que tratan sobre términos más específicos de los dominios de las mencionadas dos ramas del Derecho, el resultado final está compuesto, entonces, por 54 normativas del Derecho Ambiental y 177 del Derecho Laboral. Estos documentos fueron parcialmente evaluados por los profesionales expertos y los resultados fueron prometedores, destacando la alta reducción del número de normativas a revisar.

Cuadro 5.2: Flujo de clasificación artificial de normativas en el dominio del Derecho Ambiental.

# total de normativas	# Derecho Ambiental	# desechos peligrosos
4766	185	**54**

Cuadro 5.3: Flujo de clasificación artificial de normativas en el dominio del Derecho Laboral.

# total de normativas	# Derecho Laboral	# accidentes de trabajo
4766	1049	**177**

Por parte de los profesionales, se destaca principalmente un muy buen rendimiento con respecto a la detección artificial de normativas correspondientes al Derecho Ambiental y al Derecho Laboral.

Ahora bien, en cuanto a la detección de documentos normativos con respecto a las temáticas de desechos peligrosos y accidentes de trabajo, aunque los resultados obtenidos por dichos clasificadores no se consideraron desalentadores, ya que reducen mucho el número de documentos plausibles para poblar la matriz legal, se deben agregar otras herramientas de filtrado para mejorar la precisión de los resultados.

Capítulo 6

Conclusiones

> *"I wish it need not have happened in my time," said Frodo. "So do I," said Gandalf, "and so do all who live to see such times. But that is not for them to decide. All we have to decide is what to do with the time that is given us."*
>
> — J. R. R. Tolkien, *The Fellowship of the Ring*

El texto es uno de los principales métodos de comunicación que poseemos como seres humanos y es el protagonista indiscutible de la web actual. La información textual disponible se encuentra en constante interacción con nuestro entorno, logrando que el manejo y la extracción de información relevante para la eventual toma de decisiones dentro de enormes flujos de documentos de texto sea una tarea prácticamente imposible tanto para individuos como para organizaciones. La naturaleza no estructurada del texto, no entendible completamente por máquinas, complejiza aún más la extracción artificial de información del mismo. Las herramientas del procesamiento del lenguaje natural, principalmente los clasificadores de texto, han demostrado ser fundamentales a la hora de tratar documentos y son sumamente atractivas de emplear en diversas empresas en las cuales sus sistemas de recomendación son el corazón del servicio que ofrecen.

Particularmente, la ingeniería legal es una rama de las ingenierías con la misión de crear herramientas capaces de consumir, analizar y obtener patrones de documentos legales para asistir tanto a profesionales de la Ley como a individuos en la toma de decisión y en la búsqueda de información legal específica. Los documentos legislativos son textos escritos por representantes de una comunidad para que sus respectivos ciudadanos y organismos cumplan con las responsabilidades que se les otorga. En la actualidad, estos numerosos documentos son publicados diariamente de forma electrónica mediante distintos boletines oficiales del gobierno. Las empresas, particularmente, necesitan del continuo análisis de documentos normativos con el fin de corroborar su cumplimiento acorde a las actividades que éstas realizan y de esta manera,

prevenir posibles sanciones del Estado. La anterior actividad es la que se denomina matricería legal.

En este trabajo se ha propuesto una arquitectura conceptual, denominada Sistema de Soporte a la Ingeniería Legal (SiSIL) con el fin de asistir en tiempo real, a profesionales en la realización de diversas tareas de la ingeniería legal, en donde el flujo de información consiste en numerosas normativas publicadas periódicamente por diversos boletines oficiales del gobierno. SiSIL presenta una arquitectura base que permite el desarrollo de nuevas y diversas aplicaciones de soporte a la toma de decisión legal. Esta propuesta deja abierta la posibilidad de incluir distintas formas de análisis de normativas de mayor complejidad, tal como nuevas aplicaciones de acuerdo a las necesidades del usuario. Estas aplicaciones pueden asistir no sólo a empresas, sino también a la comunidad en general para que esta última se involucre en una Justicia más abierta e inclusiva, en el marco del programa *Justicia 2020*[1] del Ministerio de Justicia y Derechos Humanos de la República Argentina.

En esta tesina se propuso una implementación de un asistente soporte a la matricería legal. SiSIL analiza todo documento normativo recuperado de determinados boletines oficiales del gobierno y sugiere aquellos que potencialmente satisfacen la necesidad de información con respecto a una empresa previamente especificada. Para realizar este análisis, como primer instancia y con el propósito de obtener los requisitos de la empresa, el sistema de soporte y el usuario experto en el contexto de la empresa se sumergen en un diálogo exploratorio del dominio de las principales ramas del Derecho de interés para dicha empresa. De esta forma, mediante la obtención de términos más específicos del dominio de ciertas ramas, el sistema recaba información acerca de las actividades que la empresa ejecuta. El análisis empleado por SiSIL a toda normativa recuperada consta de una evaluación de contenido textual mediante la aplicación en cadena de distintos clasificadores binarios de texto con el objeto de detectar contenido relevante con respecto a términos específicos de las distintas ramas del Derecho de interés. La información extraída mediante el análisis en tiempo real de documentos normativos es utilizada, finalmente, por una aplicación de usuario en donde este último interactúa con los documentos potencialmente relevantes sugeridos por el sistema y decide la verdadera relevancia de dichos documentos. Esta aplicación de usuario, junto con el sistema diseñado para el soporte a la decisión en tiempo real a la ingeniería legal, tiene como principal finalidad asistir y alivianar la periódica tarea de profesionales encargados de la selección diaria de documentos normativos exigibles a una empresa. En esta tesina se ha logrado un primer prototipo de esta aplicación con resultados preliminares promisorios, los cuales podrán ser mejorados explorando distintas líneas de trabajo.

[1] `http://www.justicia2020.gob.ar/`

6.1. Trabajo futuro

A continuación, se enuncian diversas líneas de trabajo para un futuro refinamiento y complemento de la aplicación de soporte a la matricería legal que se propuso en la presente tesina:

- **Extracción de entidades**. Los documentos normativos son largos documentos de texto en los cuales es muy difícil para el humano identificar rápidamente términos importantes como, por ejemplo, organizaciones, fechas, regiones geográficas, nombres de personas, expresiones numéricas como dinero y números de normas a las cuales hace referencia, etc. Según (Shaalan 2014), el reconocimiento de entidades es una herramienta del procesamiento del lenguaje natural con la habilidad de clasificar entidades que están presentes en el texto en categorías predefinidas. Aplicar la anterior técnica en documentos normativos con el fin de destacar en el texto las entidades reconocidas, posiblemente transforme el proceso de evaluación de un usuario experto con respecto a la relevancia de una normativa en una tarea visual más rápida y ágil.

- **Aprendizaje en línea**. Dado que se ha planteado a la matricería legal como una actividad en tiempo real en donde el usuario brinda explícitamente su feedback cada vez que acepta o descarta un documento sugerido por el sistema propuesto, una posibilidad a explorar es emplear métodos de aprendizaje automatizado en línea como, por ejemplo, el algoritmo *Passive-Agressive* descrito en (Crammer et al. 2006).

- **Sobremuestreo de texto**. Como se ha podido notar a lo largo del trabajo, al momento de aprender modelos clasificadores de texto se sufre del problema de clases desbalanceadas. El problema es severo cuando los requerimientos del usuario son cada vez maś específicos; es decir, se posee mucha menos información positiva que negativa de la cual aprender. En estos casos, aplicar sobremuestreo de información suele solventar parcialmente esta dificultad. Pero en el dominio del texto, a comparación de otras áreas, el sobremuestreo no se considera sencillo y en los últimos años han surgido técnicas como las que se presentan en (Liu et al. 2009) y (Iglesias et al. 2013) las cuales pueden ser aplicables en el problema de la matricería legal.

- **Clasificación por organismos del Estado**. Analizando la matriz legal inicial del caso de estudio explorado en esta tesina, se ha podido observar que existen ciertos organismos del Estado, encargados de emitir ciertos tipos de normativas específicas, que poseen más frecuencia dentro de la matriz. En la Argentina existen alrededor de 200 organismos; por lo tanto, es relevante utilizar esta información con la finalidad de modelar un filtrado de documentos más refinado.

- **Clasificación basado en reglas**. Como se comentó, a medida que el usuario experto explora en profundidad los dominios de las ramas del Derecho se dispone de menos información positiva para aprender clasificadores binarios. Para solventar lo anterior, se evalúa obtener k términos más relevantes (utilizando tf-idf como esquema de pesaje en el modelo Espacio-Vectorial) de los pocos documentos disponibles, con la finalidad de crear conjuntos de reglas específicas para determinar la clase de una normativa. El antecedente de cada regla indica los términos que deben estar presente en un documento para que la misma se encienda. Se espera utilizar el algoritmo de aprendizaje *Word2Vec*, detallado en (Mikolov et al. 2013), para aumentar el antecedente de toda regla mediante términos similares a los ya especificados.

Bibliografía

Aggarwal, Charu C. (2015). *Data mining: the textbook*. Springer.

Aggarwal, Charu C. (2016). *Recommender systems*. Springer.

Aggarwal, Charu C. (2018). *Machine Learning for text*. Springer.

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

Alpaydin, Ethem (2010). *Introduction to machine learning*. MIT press.

Assal, Hisham, John Seng, Franz Kurfess, Emily Schwarz y Kym Pohl (2011). "Semantically-enhanced information extraction". En: *IEEE Aerospace Conference Proceedings*, págs. 1-14.

Baeza-Yates, Ricardo A. y Berthier Ribeiro-Neto (1999). *Modern Information Retrieval*. Vol. 463. ACM press New York.

Baggio, Bobbe Gaines (2016). *Analyzing Digital Discourse and Human Behavior in Modern Virtual Environments*. IGI Global.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer.

Borovicka, Tomas, Marcel Jirina, Pavel Kordik y Marcel Jiri (2012). "Selecting Representative Data Sets". En: *Advances in Data Mining Knowledge Discovery and Applications*. InTech.

Chen, J. J., C. A. Tsai, H. Moon, H. Ahn, J. J. Young y C. H. Chen (2006). "Decision threshold adjustment in class prediction". En: *SAR and QSAR in Environmental Research* 17.3, págs. 337-352.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz y Yoram Singer (2006). "Online passive-aggressive algorithms". En: *Journal of Machine Learning Research* 7, págs. 551-585.

Cybenko, George y Brian Brewington (1999). "The Foundations of Information Push and Pull". En: *The mathematics of information coding, extraction and distribution*, págs. 9-30.

Dale, Robert (2019). "Law and Word Order: NLP in Legal Tech". En: *Natural Language Engineering* 25.1, págs. 211-217.

Elgendy, Nada y Ahmed Elragal (2014). "Big data analytics: a literature review paper". En: *Industrial Conference on Data Mining*, págs. 214-227.

Eraso, Hugo Armando Ordoñez y Carlos Alberto Cobos Lozada (2011). "Stemming en Español para Documentos Recuperados de la Web". En: *Revista Unimar* 58, págs. 107-114.

Fawcett, Tom (2006). "An introduction to ROC analysis". En: *Pattern recognition letters* 27.8, págs. 861-874.

Felfernig, A y R Burke (2008). "Constraint-based recommender systems: technologies and research issues". En: *Proceedings of the 10th international conference on Electronic commerce*, pág. 3.

García, Salvador, Sergio Ramírez, Julián Luengo y Francisco Herrera (2016). "Big Data: Preprocesamiento y calidad de datos". En: *Novática*, págs. 17-23.

Hai Dong, Farookh Khadeer Hussain y Elizabeth Chang (2008). "A survey in traditional information retrieval models". En: *2008 2nd IEEE International Conference on Digital Ecosystems and Technologies*. IEEE, págs. 397-402.

Hastie, Trevor, Robert Tibshirani y Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer Science & Business Media.

Hussein, Doaa Mohey El Din Mohamed (2018). "A survey on sentiment analysis challenges". En: *Journal of King Saud University - Engineering Sciences* 30.4, págs. 330-338.

Iglesias, E. L., A. Seara Vieira y L. Borrajo (2013). "An HMM-based oversampling technique to improve text classification". En: *Expert Systems with Applications* 40.18, págs. 7184-7192.

James, Gareth, Daniela Witten, Trevor Hastie y Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Vol. 112. Springer.

Janssen, Marijn, Haiko van der Voort y Agung Wahyudi (2017). "Factors influencing big data decision-making quality". En: *Journal of Business Research* 70, págs. 338-345.

Japkowicz, Nathalie y Mohak Shah (2011). *Evaluating learning algorithms: a classification perspective*. Cambridge University Press.

Joachims, Thorsten (2001). "A statistical learning model of text classification for support vector machines". En: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, págs. 128-136.

Joachims, Thorsten (2002). *Learning to classify text using support vector machines*. Boston, MA: Springer Science & Business Media.

Kaplen, Ronald M. (2005). "A method for tokenizing text". En: *Inquiries into words, constraints and contexts* 55.

Kaufman, Shachar, Saharon Rosset y Claudia Perlich (2012). "Leakage in data mining: Formulation, detection, and avoidance". En: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4, pág. 15.

Kompan, Michal y Mária Bieliková (2010). "Content-based news recommendation". En: *International conference on electronic commerce and web technologies*, págs. 61-72.

Kuhn, Max y Kjell Johnson (2013). *Applied Predictive Modeling*. Springer.

Lika, Blerina, Kostas Kolomvatsos y Stathes Hadjiefthymiades (2014). "Facing the cold start problem in recommender systems". En: *Expert Systems with Applications* 41.4, págs. 2065-2073.

Lin, Wei Jiun y James J. Chen (2013). "Class-imbalanced classifiers for high-dimensional data". En: *Briefings in Bioinformatics* 14.1, págs. 13-26.

Lippi, Marco, Przemysław Pałka, Giuseppe Contissa, Francesca Lagioia, Hans Wolfgang Micklitz, Giovanni Sartor y Paolo Torroni (2019). "CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service". En: *Artificial Intelligence and Law* 27.2, págs. 117-139.

Liu, Ying, Han Tong Loh y Aixin Sun (2009). "Imbalanced text classification: A term weighting approach". En: *Expert Systems with Applications* 36.1, págs. 690-701.

Manning, Christopher D., Prabhakar Raghavan e Hinrich Schutze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Merendino, Alessandro, Sally Dibb, Maureen Meadows, Lee Quinn, David Wilson, Lyndon Simkin y Ana Canhoto (2018). "Big data, big decisions: The impact of big data on board level decision-making". En: *Journal of Business Research* 93, págs. 67-78.

Mikolov, Tomas, Kai Chen, Greg Corrado y Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". En: *Neural information processing systems*, págs. 3111-3119.

Mitchell, Thomas M. (1997). *Machine Learning*. 1.ª ed. New York, NY, USA: McGraw-Hill, Inc.

Mohri, Mehryar, Afshin Rostamizadeh y Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press.

Nath, Jitendra, Singh Sanjay y Kumar Dwivedi (2013). "A comparative study on approaches of vector space model in information retrieval". En: *International Journal of Computer Applications* 975, pág. 8887.

Neil, Martin, Norman Fenton, David Lagnado y Richard David Gill (2019). "Modelling competing legal arguments using Bayesian model comparison and averaging". En: *Artificial Intelligence and Law*.

Olszak, Celina M. (2016). "Toward Better Understanding and Use of Business Intelligence in Organizations". En: *Information Systems Management* 33.2, págs. 105-123.

Pazzani, Michael J y Daniel Billsus (2007). "Content-Based Recommendation and Systems". En: *The adaptive web*. Springer, págs. 325-341.

Pérez-Rosas, Verónica, Bennett Kleinberg, Alexandra Lefevre y Rada Mihalcea (2017). "Automatic Detection of Fake News". En:

Pilászy, István (2005). "Text categorization and support vector machines". En: *The proceedings of the 6th international symposium of Hungarian researchers on computational intelligence*. Vol. 1.

Porter, M. F. (1997). "An algorithm for suffix stripping". En: *Readings in information retrieval*, págs. 313-316.

Rajaraman, Anand. y Jeffrey David. Ullman (2011). *Mining of massive datasets*. Cambridge: Cambridge University Press.

Ricci, Francesco, Lior Rokach y Bracha Shapira (2015). *Recommender Systems Handbook*. Springer.

Robaldo, Livio, Serena Villata, Adam Wyner y Matthias Grabmair (2019). "Introduction for artificial intelligence and law: special issue "natural language processing for legal texts"". En: *Artificial Intelligence and Law* 27.2, págs. 113-115.

Russell, Stuart J y Peter Norvig (2016). *Artificial intelligence: a modern approach*. Pearson Education Limited.

Ruthven, Ian y Mounia Lalmas (2003). "A survey on the use of relevance feedback for information access systems". En: *The Knowledge Engineering Review* 18.2, págs. 95-145.

Salton, G., A. Wong y C. S. Yang (1975). "A vector space model for automatic indexing". En: *Communications of the ACM* 18.11, págs. 613-620.

Schafer, J. Ben, Dan Frankowski, Jon Herlocker y Shilad Sen (2007). "Collaborative filtering recommender systems". En: *The adaptive web*. Springer, págs. 291-324.

Sebastiani, Fabrizio (2002). "Machine learning in automated text categorization". En: *ACM computing surveys (CSUR)* 34.1, págs. 1-47.

Shaalan, Khaled (2014). "A Survey of Arabic Named Entity Recognition and Classification". En: *Computational Linguistics* 40.2, págs. 469-510.

Shanahan, James G. y Norbert Roma (2003). "Improving SVM Text Classification Performance through Threshold Adjustment". En: *European Conference on Machine Learning*. Springer, págs. 361-372.

Shardanand, Upendra y Pattie Maes (1995). "Social information filtering: algorithms for automating Word of Mouth". En: *Chi*, págs. 210-217.

Shimazu, Akira y Minh Le Nguyen (2014). "Legal Engineering and Its Natural Language Processing". En: *Knowledge and Systems Engineering*. Springer, págs. 7-7.

Singh, Sonit (2018). "Natural Language Processing for Information Extraction". En: págs. 1-24.

Tharwat, Alaa (2018). "Classification assessment methods". En: *Applied Computing and Informatics*.

Waltl, Bernhard, Georg Bonczek, Elena Scepankova y Florian Matthes (2019). "Semantic types of legal norms in German laws: classification and analysis using local linear explanations". En: *Artificial Intelligence and Law* 27.1, págs. 43-71.

Wang, Hai, Zeshui Xu, Hamido Fujita y Shousheng Liu (2016). "Towards felicitous decision making: An overview on challenges and trends of Big Data". En: *Information Sciences* 367, págs. 747-765.

Yamada, Hiroaki, Simone Teufel y Takenobu Tokunaga (2019). "Building a corpus of legal argumentation in Japanese judgement documents: towards

structure-based summarisation". En: *Artificial Intelligence and Law* 27.2, págs. 141-170.

Yu, Liyang (2011). *A developer's guide to the semantic Web*. Springer Science & Business Media.

Yu, Wei, Tiebin Liu, Rodolfo Valdez, Marta Gwinn y Muin J. Khoury (2010). "Application of support vector machine modeling for prediction of common diseases: The case of diabetes and pre-diabetes". En: *BMC Medical Informatics and Decision Making* 10.1, pág. 16.

Yuan, Guo Xun, Chia Hua Ho y Chih Jen Lin (2012). "Recent advances of large-scale linear classification". En: *Proceedings of the IEEE* 100.9, págs. 2584-2603.

Zhang, Haiyi y Di Li (2007). "Naive Bayes text classifier". En: *2007 IEEE International Conference on Granular Computing (GRC 2007)*, págs. 708-711.