UNIVERSIDAD SANTO TOMÁS

MODELADO DE UN ALGORITMO PARA LA IDENTIFICACIÓN DE PROBLEMAS JURÍDICOS Y TESIS EN LAS PROVIDENCIAS DE LA SECCIÓN PRIMERA DEL CONSEJO DE ESTADO USANDO TÉCNICAS DE PROCESAMIENTO DEL LENGUAJE NATURAL

Realizado por

SERGIO MAURICIO SALAZAR MOLINA

WILLIAM ANDRÉS PINTO CÁCERES

Facultad de Ingeniería Electrónica

División de Ingenierías

Julio de 2022

MODELADO DE UN ALGORITMO PARA LA IDENTIFICACIÓN DE PROBLEMAS JURÍDICOS Y TESIS EN LAS PROVIDENCIAS DE LA SECCIÓN PRIMERA DEL CONSEJO DE ESTADO USANDO TÉCNICAS DE PROCESAMIENTO DEL LENGUAJE NATURAL

Realizado por

SERGIO MAURICIO SALAZAR MOLINA
WILLIAM ANDRÉS PINTO CÁCERES

Dirigido por

Juan Manuel Calderón Chávez

Codirigido por

ANA YASMÍN TORRES TORRES

Facultad de Ingeniería Electrónica
División de Ingenierías

Julio de 2022

DEDICATORIA

Este proyecto lo dedicamos a nuestros familiares, quienes siempre nos apoyaron y estuvieron con nosotros a lo largo del desarrollo y ejecución del mismo. Por siempre darnos compañía, motivación, felicidad y comprensión en todo este tiempo. También a nuestros docentes y directores quienes nos guiaron, y siempre estuvieron preocupados por nuestro estado y bienestar.

Gracias a todos de la manera más sincera.

Sergio Salazar y William Pinto

AGRADECIMIENTOS

Este trabajo esta dedicado a mi padre, Luis Felipe Molina Guerrero, mi héroe, mi luz, mi guía, que su apoyo incondicional me hizo crecer como una persona honesta y dedicada. Gracias por protegerme y por creer en mi, no me alcanzaría la vida para agradecerte todo lo que has hecho por mi, has sido la persona mas importante de mi vida y así será siempre, aunque hoy estés lejos, solo abrázame con el pensamiento y seré feliz. También está dedicado a mi madre que ha sido mi soporte más grande y la razón para seguir adelante.

Quiero agradecer de manera especial a Juan Manuel Calderón y Sindy Paola Amaya, quienes sin su guía no habría sido posible la realización de este trabajo.

Gracias, gracias, gracias.

Sergio Mauricio Salazar Molina

Estos agradecimientos no son para muchas personas, pero la sinceridad con la cual ellos dedicaron su tiempo y con el cual nos apoyaron es algo que no se puede agradecer en este simple párrafo. Es por esto que quiero dar un agradecimiento a mi familia, más concretamente a mi madre quien siempre estuvo ahí para mí, siempre buscaba que trabajara con una sonrisa y que siempre esperaba lo mejor para mí. De la misma forma quiero agradecer a los directores del proyecto, por dar más de lo que deberían para lograr este proyecto en cada una de las etapas, buscando siempre un buen resultado. De la manera más sincera agradezco todo lo que dieron y espero que el mundo siga teniendo personas tan grandiosas como ustedes.

William Andrés Pinto Cáceres

Índice general

Índice de Figuras

Índice de Tablas

Glosario

BOW:(*Bow of Words*) es un método de simplificación que se utiliza en el procesado del lenguaje para describir la ocurrencia de palabras en un documento. Con este modelo se puede medir la presencia de palabras conocidas en un documento.

CFBF:*(Compound File Binary Format)* es un formato de archivo de documento compuesto desarrollado por *Microsoft* utilizado para almacenar numerosos archivos y secuencias dentro de un solo archivo en un disco.

Corpus: Base de datos compuesta por documentos de texto.

Aprendizaje profundo: *(Deep learning)* Es un conjunto de algoritmos inspirados en la estructura y función del cerebro humano para enseñarle a las máquinas a procesar información.

DIFAT: (*Double-Indirect FAT*) Es una estructura que se utiliza para ubicar sectores *FAT* en un archivo compuesto. **FAT:** (*File Allocation Table*): es un sistema de archivos desarrollado para *MS-DOS*.

FAT32: Es un sistema de archivos desarrollado como una mejora de *FAT16* y del antiguo *FAT* creado para *MS-DOS*.

FASTTEXT: Es una librería desarrollada para el aprendizaje de incrustaciones de palabras y clasificación de texto, creada por el laboratorio de investigación de inteligencia artificial (*FAIR*) de *Facebook*.

FIB:(*File Information Block*) Es el encabezado de un archivo de *Word*. Comienza en el desplazamiento 0 en el archivo y proporciona el desplazamiento inicial y las longitudes del flujo de texto del documento y las estructuras de datos subsidiarias dentro del archivo.

ID: Identificador.

Aprendizaje automático: *(Machine learning)* Es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. [1]

ODF: (*Open Document Format*) Es un estándar de documentos de datos completamente abierto y estandarizado por la *ISO*, cuyo objetivo es permitir que los documentos creados con cualquier aplicación ofimática puedan ser interpretados correctamente por otra.

OLE: Los documentos *OLE*, son sistemas de objetos distribuidos y entrelazados para ser tratados en otros documentos.

OOXML: (*Office Open XML*) Es un formato basado en *XML* utilizado para almacenar diferentes tipos de información, tales como gráficas y hojas de calculo en un documento de texto tipo *office*.

PDF:(*Portable Document Format*) Es un formato de almacenamiento para documentos creado por Adobe. Este tipo de formato es utilizado para intercambiar documentos independientes de *software* y *hardware* sin que cambie el formato del documento en cuestión.

PLN/NLP: (Procesamiento del Lenguaje Natural o *Natural Language Processing*) Es una rama de la inteligencia artificial y de la lingüística, enfocada a procesar e interpretar textos en lenguaje humano por medio de la transformación al lenguaje maquina y la aplicación de metodologías de aprendizaje automático.

TEXT CNN: (*Convolutional neural network for text*) Es un algoritmo de aprendizaje profundo utilizado para tareas de clasificación de oraciones, como el análisis de sentimientos y la clasificación de preguntas.

Tokenización: Separar y/o unificar caracteres, palabras u oraciones. Estos conjuntos se les denomina tokens.

XML:(*Extensible Markup Language*) Es un metalenguaje diseñado para almacenar, transmitir, y reconstruir datos de forma legible.

Word2Vec: Es un método del Procesamiento del Lenguaje Natural utilizado para aprender asociaciones de palabras de un gran corpus de texto de manera eficiente mediante el uso de una red neuronal.

Resumen

El Consejo de Estado es el órgano de cierre y máximo Tribunal de lo Contencioso Administrativo en Colombia. En el desarrollo de sus funciones, los consejeros profieren decisiones denominadas autos, sentencias o conceptos. En firme sus decisiones, son remitidas a la relatoría de la corporación para su análisis, estandarización, titulación y sistematización. De esta manera quedan disponibles al público para su consulta. La labor de titulación requiere un equipo de trabajo, encargado de incluir en el sistema los datos generales de las providencias, rotularlas a través de palabras claves (descriptores y restrictores), extraer la tesis y los aspectos que consideren relevantes para la debida clasificación y posterior consulta de las decisiones.

Este proyecto se dividió en dos etapas. En la primera etapa busca reducir la labor mecánica de estandarización de la información plasmada en los diferentes documentos (providencia, autos, fallos) que adelantan los relatores, mediante el uso de técnicas de Aprendizaje de Máquina, toda vez que desde la Inteligencia Artificial (IA), estos datos se pueden extraer de forma automática y en menores periodos de tiempo a los utilizados por los relatores. Esto le permitirá al equipo de Relatoría ejercer las labores jurídicas propias de su cargo, es decir, el análisis y estudio de las decisiones de la corporación. En el enfoque del Marco Lógico, el problema central de la investigación reside en la dificultad que presenta el consultar el material de la Relatoría del Consejo de Estado, debido a la variedad de términos y estilos de redacción en los que estos documentos son generados y almacenados. Dadas las circunstancias, la ejecución de un sistema de automatización del proceso de titulación de las relatorías, facilitara el acceso al público a la documentación del Consejo de Estado, por medio de la clasificación y estandarización de los documentos.

En la segunda etapa se busca automatizar parte del proceso de identificación y extracción de las tesis y los problemas jurídicos de las sentencias de la Sección Primera del Consejo de Estado. Esta labor se plantea mediante un Sistema Inteligente (IS) basado en Procesamiento del Lenguaje Natural (NLP) para el análisis e interpretación de las sentencias, que facilite de forma eficaz y eficiente, la comprensión y la extracción automática del Problema Jurídico y la Tesis.

Introducción

El proceso de la titulación es un proceso realizado por la Relatoría para generar la síntesis de las providencias que se desarrollan dentro del Consejo de Estado, sin embargo, este proceso se llega a resumir en la extracción literal de los problemas jurídicos y las tesis de estos documentos, el cual los relatores han desarrollado de manera manual durante años y que poco a poco se van atrasando en esta tarea ante el crecimiento de la población en el país y el número de casos a procesar, como se menciona en el capítulo 1. Es por estas razones que nace este proyecto, el cual ha tomado origen ante la necesidad de automatizar este proceso mecánico realizado por profesionales que están capacitados para realizar otras labores en el ámbito de lo jurídico.

Para dar solución a este problema se ha optado por la aplicación de un algoritmo desarrollado con técnicas de Procesamiento del Lenguaje Natural, gracias a su capacidad de igualar este proceso de análisis y extracción textual, además de las generaciones del formato estándar para las titulaciones. Este algoritmo se describe en el capítulo 2, donde se hace referencia a las distintas aplicaciones que se han desarrollado de la inteligencia artificial y el Procesamiento del Lenguaje Natural a través de los años y los países para diferentes documentos legales.

Para el desarrollo del proyecto, se recolectó la información relacionada al proceso de titulación descrito en el Manual de la Relatoría y la descripción verbal de la labor realizada por los relatores del Consejo de Estado. Además, se ha indagado sobre los procesos relacionados para el Procesamiento del Lenguaje Natural, así mismo sobre las reglas a seguir para dar formato a los documentos, recolectando esta información en el capítulo 6. Este desarrollo se ha redactado con detalle en el capítulo 7, donde el algoritmo comienza con la realización de la base de datos, la cual estará compuesta por los documentos sin procesar. Posteriormente se procederá a transformar los párrafos de cada uno de los documentos, en representaciones vectoriales de las ocurrencias de cada una de las palabras. Una vez completados los pasos anteriores, se procede a dar paso a la aplicación los métodos de Procesamiento del Lenguaje Natural en cuestión. Esta aplicación estará orientada al análisis del contexto de los documentos legales que harán parte del corpus, para extraer aquellos párrafos que

representen el problema jurídico y la tesis de la providencia y descartar la información que es irrelevante para la generación de las titulaciones.

Es de esta manera que en el presente documento se dará a detalle el desarrollo, los resultados y los alcances de un algoritmo desarrollado para la generación de titulaciones de la Sección Primera del Consejo de Estado por medio de técnicas de Procesamiento del Lenguaje Natural.

Este proyecto de grado fue desarrollado bajo la dirección y apoyo del Grupo de investigación y Desarrollo en Robótica de la Universidad Santo Tomas (G.E.D.). Este grupo de investigación se ha enfocado principalmente en tres áreas de investigación denominadas Robótica Móvil, Visión Artificial y Sistemas Inteligentes. Durante los últimos años el G.E.D. ha desarrollado proyectos de investigación enfocados en diferentes áreas de robótica tales como robótica de enjambre, robótica cooperativa, desarrollo de robots móviles, sistemas de aprendizaje automático y robótica educativa entre otros. En el área de robótica de enjambre se han desarrollado algoritmos de navegación de enjambres [2], [3] y generación de comportamientos colectivos [4], [5], [6]. Se desarrollaron proyectos encaminados al diseño e implementación de plataformas robóticas móviles para la evaluación de comportamiento de sistemas multi-agente [7] y robots de asistencia casera [8], [9]. En al área de los sistemas de aprendizaje se han realizado proyectos encaminados al uso de sistemas de aprendizaje para la generación de comportamientos específicos en la navegación de robots móviles [10] y también para sistemas de recomendación de estilos de vestir [11]. Adicionalmente, el grupo ha participado activamente de la iniciativa RoboCup [12], como parte de la liga Small Size [13], [14] y la generación de iniciativas educativas encaminadas a la popularización de la robótica a nivel de educación secundaria [15] y universidad [16].

Capítulo 1

Problema

El Consejo de Estado es el máximo juez de la administración pública para la resolución de conflictos entre las entidades estatales y los ciudadanos, o así mismo entre entidades del propio estado, el cual también se encarga de asesorar al Gobierno Nacional en asuntos que requieran una atención especial dada la trascendencia del problema, como se estipula en el artículo 237 de la Constitución Política de Colombia, como órgano de cierre tiene competencia ante todo el territorio Colombiano, es por esta razón que todas las decisiones y conceptos adoptados dentro de este órgano tienen un impacto en todas las áreas del territorio nacional. Una vez comunicadas a los interesados las decisiones y conceptos adoptados por los directores, se remiten las decisiones a la Relatoría para que los relatores cumplan su labor de identificar, extraer y titular las tesis y los problemas jurídicos de las jurisprudencias correspondientes de cada sección. Este trabajo es realizado por 48 personas que dedican la mayor parte de su tiempo a completar los campos requeridos por el sistema, incluyendo descriptores y calificadores (palabras claves), dando gran prioridad a labores mecánicas en vez de enfocarse en sus funciones jurídicas, todo con el objetivo de que diferentes usuarios, tanto como la ciudadanía, así como los jueces puedan consultar y descargar las providencias, ante un sistema que además carece de un formato en concreto. Según la Coordinación de Relatores del Consejo de Estado, en 2019 se titularon 16.842 órdenes y conceptos, y en 2020 van 9.162 títulos. Las decisiones tituladas son consultadas por toda la comunidad jurídica y por juristas a nivel nacional e internacional. Según el buró de sistemas, 2.283.095 usuarios consultaron las jurisprudencias en 2019, y en 2020 se han realizado 91.216 consultas hasta el momento.[17]

Actualmente en lo que respecta al país, se busca el desarrollo social y económico por medio de la industria 4.0, siendo uno de los principales aceleradores, el desarrollo de la inteligencia artificial; al ser uno de los mayores impulsores de un gran sector, permite transformar una

gran variedad de procesos que usualmente son realizados de manera mecánica por humanos al entorno digital, siendo de gran ayuda ante un mundo que cada año exige una mayor manipulación de datos [17], siendo el sector jurídico uno de los sectores que más empieza a presentar dificultades ante la organización [18], clasificación y consulta de los diversos problemas jurídicos que día a día se presentan, es por ello que la respuesta que han dado diferentes países en la última década ante este problema ha sido la implementación por medio de técnicas de Aprendizaje automático, Aprendizaje profundo o PLN (Procesamiento del Lenguaje Natural) en pequeños sectores ante la grandeza de esta estructura y para evitar afectar drásticamente el sistema judicial, como viene a ser el análisis de evidencias respecto a su veracidad, la posible toma de decisiones basadas en casos previos y la clasificación de documentos según su cuerpo textual [19]. Es debido a estos factores que se desea una implementación dentro de las relatorías del Consejo de Estado del país, por lo que se genera la pregunta:

¿Cómo extraer y clasificar automáticamente los problemas jurídicos y las tesis en las providencias de la Sección Primera del Consejo de Estado mediante el uso del procesamiento de lenguaje natural?

Capítulo 2

Antecedentes

Las funciones que debe desempeñar el informante, así como la estructura y tramitación de las sentencias procesadas en el ámbito legal están estipuladas en el reglamento interno del Consejo de Estado (Acuerdo No. 080 de 2019) [20] más específicamente en el artículo 61 y artículo 62. El informe señala claramente que la Relatoría debe asegurar que la clasificación, organización y los correspondientes registros en la base de datos de la sede electrónica del Consejo de Estado se implementen de la mejor manera, resolviendo en la organización la finalidad, conceptos, toma de decisiones y cuestiones judiciales. El sitio web proporciona usabilidad y accesibilidad para el público.

El análisis del lenguaje ya había sido usado en el ámbito legal, jurídico y la criminología de manera manual, pero ante el crecimiento de la sociedad, dejó de ser un factor rentable [21]. Su transformación al mundo digital se ha realizado a través de la inteligencia artificial en gran medida [22]. Un ejemplo es el caso mencionado en el artículo "*Classifying Semantic Types of Legal Sentences: Portability of Machine Learning Models*" [23], en este artículo se hace uso de distintos métodos de aprendizaje automático para la clasificación de contratos legales, sin embargo en el ámbito legal de los contratos con lo que respecta al idioma Alemán, existe un gran nivel de dificultad en el acceso de estos por temas de privacidad, es debido a ello que la investigación se orientó en analizar la capacidad de portabilidad de modelos de Aprendizaje automático por medio del aprendizaje de documentos diferentes al deseado, pero dentro del área del estudio de lo legal[24].

Principalmente se ha visto un interés para la ampliación de la inteligencia artificial dentro del entorno jurídico[25], ha sido en la toma de decisiones, sin embargo, no llegue a utilizarse para un factor absolutamente decisivo ante la desconfianza [26] y el dilema que aún existe de dejar

temas legales totalmente a máquinas, otro caso es el mencionado en "*Using machine learning to predict decisions of the European Court of Human Rights*" [27] donde investigadores europeos por medio del procesamiento natural del lenguaje y haciendo uso de diversos textos judiciales referentes a casos en los que se habían visto vulnerado los derechos humanos[21], según lo estipulado en el Convenio Europeo de Derechos Humanos, han desarrollado un sistema basado en el aprendizaje automático, para la predicción automática de los veredictos, logrando en general una precisión del 75 % en las predicciones.

El análisis de los documentos se desarrolló para tres diferentes temáticas, la demostración de la aplicabilidad del Aprendizaje automático en el entorno jurídico eliminando durante la etapa del aprendizaje principalmente toda información relacionada con el veredicto (Argumentos que llevaron a ello y la decisión tomada)[28], también se analizó su capacidad ante futuros casos alterando la fecha de los documentos analizados para simular una evaluación más real ante el constante cambio por el cual pueden pasar las leyes a través del tiempo, lo cual resultó en una precisión entre el 58 % y 68 % y finalmente se analizó su capacidad de predicción en base al nombre juez que se hacía cargo del caso demostrando en gran medida la influencia de este factor al obtener una precisión del 65 % .

Un caso similar al anterior es mencionado en "*A Deep Learning Method for Judicial Decision Support*" [28] donde haciendo uso de métodos de Aprendizaje profundo y NLP construyen diferentes modelos basados en FastText y TextCNN para la predicción de penalizaciones, disposiciones legales y acusaciones, con el objetivo de servir de referencia en la toma de decisiones y reducir la brecha que pueda existir ante casos similares, otro ejemplo del trabajo en la toma de decisiones es "*A Markov logic networks based method to predict judicial decisions of divorce cases*"[25] donde en este proyecto se toma aquello relacionado a casos de divorcios pero por medio de redes lógicas de Márkov.

De los últimos desarrollos que se han realizado del uso de la inteligencia artificial en el entorno jurídico es el mencionado en la conferencia de "*Similarity Analysis of Law Documents Based on Word2vec*" [29] donde se propuso el uso de la técnica de Procesamiento del Lenguaje Natural, Word2Vec, el cual utiliza un modelo de red neuronal para aprender sobre las similitudes de palabras en documentos generalmente extensos, siendo para este caso el objetivo de aplicarla en documentos legales, sin embargo una de las mayores dificultades que llega a presentar el desarrollo para el análisis de la similitud es la diferencia de los formatos entre los documentos y la extensión de estos, lo cual es una de las principales problemáticas para la aplicación en los documentos legales debido a su gran diversidad de formatos ante la carencia de uno en concreto y la extensa longitud que pueden llegar a tener sobre un solo caso. Es en este documento donde se llega a analizar la eficiencia del método de Word2Vec para

documentos legales, comparándolo con el modelo de bolsa de palabras (BOW) y comparando la similitud obtenida por diferentemente modelos de Word2Vec, siendo esta calculada por medio de la similitud coseno. Los resultados del artículo dejan en claro su posibilidad para la aplicación en documentos legales además de que se puede desarrollar a mayor medida, sin embargo, consume demasiados recursos del sistema y consume mucho tiempo para su formación [30].

Por otro lado, mirando en el ultimo año, entre las investigaciones que se han presentado está la del articulo "*Deep Learning Techniques for Legal Text Summarization*", presentado en la conferencia Internacional UPCON. En este documento se detalla la investigación realizada para la posible aplicación de redes neuronales en la ejecución de resúmenes de documentos legales, por medio de técnicas del PLN. Dado los tiempos que consume esta labor, junto a la alta necesidad, la dificultad que suele representar su ejecución y el alto costo de profesionales en el área de derecho calificados para esta función. La investigación concluye que es sugerible la elaboración de resúmenes híbridos, con características abstractas y extractivas para mantener la información original y su legibilidad, dando enfoque al uso de transformadores [31].

Otra investigación realizada el último año corresponde al artículo "*Analysis of Legal Case Document Automated Summarizer*", presentado en la ISPCC, donde se investiga las técnicas para el resumen de documentos legales posiblemente aplicables en el sistema legal de la India. Esta investigación se realizó con el objetivo de reducir tiempo en la labor, sin el objetivo de suplantar la labor de los abogados, por medio de la extracción de los casos menos complejos. En este artículo se hace un análisis sobre las técnicas basadas en el PLN. Posteriormente, realiza la aplicación y la evaluación con sistemas ya planteados para otros países. Entre los sistemas probados están CaseSummarizer previamente desarrollado para documentos legales australianos, el enfoque LetSum desarrollado para documentos canadienses y un modelo grafico (CRF). Donde finalmente quedaron abiertos a más investigación a detalle, dado que los sistemas no fueron capaces de dar con la razón de la realización del caso y sus fallos [32].

En lo que respecta a los desarrollos del último año, se ha logrado el desarrollo de modelos como el que se menciona en el artículo "*LegalDB: Long DistilBERT for Legal Document Classification*". En este artículo se realiza la clasificación de documentos legales de los Estados Unidos dentro de 4 categorías (Sentencias de la Corte Suprema y de Circuito, reclamo federal, apelación fiscal y declaración de bancarrota) utilizando un modelo de transformadores DistilBERT. En este articulo se propone dar solución a las limitaciones de la longitud de los contextos y el tamaño de los modelos, por medio de una nueva arquitectura, para los casos de los modelos de codificación automática basados en transformadores. Dadas las limitaciones

para documentos extensos y con vocabulario alejado del vocabulario general del idioma. La solución se da utilizando atención global y local en DistilBERT, basado en el concepto de atención deslizante de Longformer, y realizando un preentrenamiento adicional y un posterior ajuste del modelo [33].

Otro de los desarrollos del último año se menciona en el artículo "*Kodiak@ALQAC2021: Deep Learning for Vietnamese Legal Information Processing*" donde se trata a dar solución por medio del aprendizaje profundo a las tareas de búsqueda y filtrado de documentos legales, la tarea de conocer la vinculación entre textos legales y la tarea de dar respuesta a preguntas de índole legal para el entorno de las leyes estatutarias vietnamitas y tailandesas. Dando solución a la primera tarea por medio de la combinación de un modelo de similitud léxica y un modelo de aprendizaje profundo. Para la segunda tarea desarrolla un modelo Multilingual-Bert por medio de dos etapas, una de preentrenamiento y una de Fine-Tuning con truncamiento asimétrico. Y para la ultima tarea por medio de la combinación de los resultados anteriores [34].

Finalmente, de los últimos desarrollos del año pasado corresponde al artículo "*Automation System Based on NLP for Legal Clinic Assistance*" desarrollado por los autores del presente documento. En este articulo se busca agilizar al proceso de consulta del Consultorio Jurídico de la Universidad Santo Tomas y ampliar la cobertura de esta labor abriendo paso a su desarrollo virtual. Esta labor se logra por medio de un sistema enfocado en clasificar las solicitudes de los ciudadanos que se registran de manera virtual en el consultorio dentro de 5 categorías de índole legal. (Civil, administrativo, laboral, criminal y de seguridad social). Para ello el sistema se desarrolló por medio del análisis, ejecución y evaluación de 5 modelos de aprendizaje automático realizados por medio de técnicas de PLN [35].

Capítulo 3

Justificación

El Consejo de Estado es uno de los entes judiciales encargados de asegurar que el desarrollo del país sea orientado a un Estado Constitucional y un Estado Democrático, sin embargo, las necesidades sociales indican que es necesario mejorar los estándares judiciales en general para lograr este objetivo [36]. Ante aquellas necesidades, se puede entender que tarde o temprano la Relatoría como cualquier ente o proceso que se haya desarrollado en el servicio de lo público llegará a un punto dónde será incapaz de suplir sus servicios ante el crecimiento y desarrollo de la sociedad [37] [38], debido al aumento del número de procesos judiciales que el Consejo de Estado debe manejar. Dado el aumento de la cantidad de procesos judiciales, se ha optado por la transformación digital y al desarrollo tecnológico para poder seguir avanzando y ejerciendo sus funciones. Es imperativo que el sector jurídico empiece a desarrollar esta transformación hacia lo digital, para dejar de realizar labores mecánicas, ya que este personal está altamente capacitado permitiéndoles ejercer otras labores jurídicas propias de su cargo. Se espera que la automatización de algunos procesos propios del Consejo de Estado ayude a descongestionar la justicia colombiana y brindar un mayor y mejor acceso al público en general a las decisiones del Consejo de Estado.

Capítulo 4

Impacto Social

En más de los 200 años que ha existido el Consejo de Estado, esté siempre ha sido y será uno de los entes más importante del país, debido a su papel en la defensa de la institucionalidad, los derechos y libertades públicas, además del equilibrio de poderes, siendo este una de las bases más esenciales para la democracia y el Estado Social de Derecho, es en ello donde la aplicación de la inteligencia artificial en lo que respecta a la administración pública se vuelve un factor esencial.[39]

Con el desarrollo e investigación de la tecnología de la inteligencia artificial y el Procesamiento del Lenguaje Natural dentro de este entorno jurídico [40], significa un beneficio en lo que a recursos y tiempos se refiere para la administración de los datos que se manejan en la relatoría, más sin embargo esto no se llega a traducir en la sustitución total del papel de los relatores, ya que la inteligencia artificial no es capaz de igualar los procesos cognitivos y autónomos que requieren esta labor, sino que se realiza para servir de apoyo ante un proceso que puede ser mecanizado y el cual carece de un formato [41] permitiendo un mejor desarrollo para futuros casos que requieran esta documentación y así generar sentencias mejor estructuradas y de mayor calidad al tener una forma estandarizada.

Debido a estos factores se genera una barrera que obstaculiza el acceso a los ciudadanos a la información que de por sí es de acceso público, pero si se dificulta encontrarla hace de esta una labor muy dispendiosa para el usuario común. Este cambio va a ser de gran ayuda para acercar, hacer público y entendible en mejor medida para la sociedad la información que ya se encuentra almacenada en las bases de datos, sobre lo que se desarrolla dentro del Consejo de Estado, además de permitirle centrarse en mayor medida a su funcionamiento en beneficio de la sociedad.

Capítulo 5

Objetivos

5.1. Objetivo General:

Diseñar un algoritmo para la identificación y extracción de los problemas jurídicos y las tesis de las decisiones dadas en la Sección Primera del Consejo de Estado mediante el uso de técnicas de inteligencia artificial.

5.2. Objetivos Específicos:

- Analizar las jurisprudencias de la Sección Primera del Consejo de Estado para encontrar una estructura general, que sirva para la generación de un formato estándar.
- Estudiar las diferentes técnicas de Procesamiento del Lenguaje Natural que puedan ser utilizadas para la identificación y extracción de los problemas jurídicos y las tesis en las jurisprudencias.
- Implementar una base de datos basada en jurisprudencias que han sido previamente tituladas por los relatores del Consejo de Estado para realizar procesos de entrenamiento y evaluación del algoritmo.
- Desarrollar un algoritmo mediante técnicas de Procesamiento del Lenguaje Natural para la identificación y extracción de los problemas jurídicos y las tesis en las jurisprudencias.
- Evaluar el funcionamiento del algoritmo para la identificación y extracción de los problemas jurídicos y las tesis en las jurisprudencias.

Capítulo 6

Marco Teórico

Las investigaciones en inteligencia artificial de los últimos años han tomado como objetivo de estudio los aspectos jurídicos, las estructuras de razonamiento legal, el análisis de texto, y la relación entre estos dos constituyen tendencias claras en la materia como lo indica Ashley [42];así mismo, Raaijmakers [43] identifica el estudio de requisitos jurídicos como un campo de interés para la investigación en inteligencia artificial. El objetivo que busca la inteligencia artificial en las áreas del derecho, consiste en lograr implementar técnicas matemáticas y computacionales para hacer la ley más entendible, manejable, útil y accesible. Así mismo se podría decir que busca hacer la ley un poco más predecible, así como lo explican los formalismos matemáticos en el ámbito del derecho. Según Leibniz estos formalismos buscan formar sistemas basados en reglas y representaciones del conocimiento modelando los argumentos legales de forma que sea computacionalmente procesable [44] .

La inteligencia artificial en la práctica del derecho se suele utilizar más que nada para la obtención de evidencia relevante entre documentos y separar características claves dado que en el ámbito del derecho se utilizan una gran cantidad de documentos. Estos procesos sobrepasan la capacidad humana de procesar dicha cantidad de datos, a lo cual, las funciones que busca reemplazar la inteligencia artificial son todas aquellas labores mecánicas y repetitivas que realizan los trabajadores del área legal, tales cómo son la predicción de respuestas legales en los veredictos o las titulaciones de providencias judiciales [19], [27], [45].

Muchos investigadores han abordado y trabajado las estructuras computacionales que reflejan las características y procesos jurídicos que, como afirman Zeleznikow y Stranieri [46][47], extraen datos, los estructuran y los asocian con procesos necesarios sobre los mismos para poder desarrollar tecnologías de inteligencia artificial.

Dado que la mayoría de los problemas jurídicos consideran la pregunta que se plantea como tesis del argumento jurídico dada una providencia, una gran parte de los estudios se centran más que nada en la extracción de información, para luego ser clasificada y categorizada de manera que pueda ser utilizada por una máquina para realizar la misma labor que realiza el abogado o el administrador de la ley, en el cual se considera el problema jurídico y la sala de decisión a partir de los hechos del caso. La tesis es la respuesta dada al problema jurídico explícita o implícitamente planteado en la sentencia, auto, concepto o decisión de definición de conflictos de competencias administrativas y que da solución al caso.[48]

6.1. Estándar de facto de formatos de archivos de documentos

Los formatos de archivos nativos de Microsoft Word se indican mediante las extensiones de archivo DOC o DOCX, siendo DOCX el nuevo estándar internacional de formato abierto para documentos de texto tipo *office* definido en *Office Open XML*.[18]

Esta clase de documentos contienen múltiples archivos basados en XML sorteados en diferentes directorios comprimidos, de forma similar a como funciona un archivo comprimido en ".ZIP". Dentro de cada directorio, se encuentra definida una secuencia denominada *"Word Document"*, la cual define mediante un bloque de información de archivo *(FIB, File Information Block)*, dónde comienza y terminan las diferentes secciones en un documento *Word*. Este archivo define el primer punto de referencia para ubicar todas las propiedades del documento, incluyendo la versión en la cual fue creado , el autor y otros atributos ajenos al texto propio del documento.

Previo al proceso de estandarización de formato de los documentos *Word*, *Microsoft Office* almacenaba sus datos mediante archivos binarios, esto provocaba una incorrecta visualización del documento para versiones previas a *Office 2007* y por consiguiente una difícil edición de los mismos.

Rich Text Format (RTF), fue uno de los primeros esfuerzos realizados para crear un formato que permitiera intercambiar texto enriquecido entre aplicaciones, como formato opcional para Word. Este formato permite conservar la mayor parte del formato del documento y todo el contenido del mismo [21].

La extensión antecesora a .DOCX, .DOC utiliza una estructura de archivos CFBF (Compound File Binary Format) similar al formato de almacenamiento FAT. En este tipo de formato, el documento es dividido en sectores encadenados junto a una tabla de asignación de archivos, que contiene los sectores relacionados para cada uno de los archivos.

Hay varios tipos de sectores que pueden estar presentes en un CFBF, entre ellos están:

- Sector de tabla de asignación de archivos (FAT): contiene cadenas de índices asociadas con clusters de almacenamiento de información.
- Sectores MiniFAT: Es una simplificación del sistema FAT que conserva el mismo concepto original del sistema FAT, pero en cadenas de mini-sectores .
- Sector FAT doblemente indirecto (DIFAT): contiene cadenas de índices utilizadas para ubicar múltiples sectores FAT en un archivo compuesto (CFBF).
- Sector de directorio: Sistema de archivos separado por directorios.
- Stream Sector: Sistema de archivos ordenado de manera arbitraria, puede contener directorios.

Como ya se ha mencionado los documentos de extensión .DOC trabajan por medio de archivos con formato XML, sin embargo, internamente sus elementos textuales, el documento principal, sus tablas, sus imágenes y sus secciones se llegan a componer como conjuntos con diferencias de formatos. Cada elemento dentro de estos documentos ya sea los encabezados, pie de páginas, notas, tablas o listas, llegan a estar conformados principalmente por párrafos dentro de lo que le corresponde en su jerarquía. Es dentro de los párrafos del documento principal, que se puede ver en mayor medida esta diferencia de formato debido a que las líneas vacías, no llegan a ser discriminadas de este conjunto de párrafos, sino que llega a contar como tal y se convierten en el indicativo de un nuevo párrafo. Además, internamente los propios párrafos llegan a ser el resultado de la concatenación de otro tipo de conjuntos denominados runs, que se dividen por el conjunto de palabras consecutivas con un mismo formato, que componen cada párrafo, diferenciándose por propiedades como la negrilla, cursiva, el subrayado, etc.

6.2. Anatomía de un archivo WORDPROCESSINGML

Un archivo WordprocessingML o DOCX es un archivo zip (un paquete) que contiene una serie de "partes", generalmente archivos XML codificados en UTF-8 o UTF-16, aunque estrictamente definido, una parte es un flujo de bytes. El paquete también puede contener otros archivos multimedia, como imágenes y vídeo. La estructura está organizada de acuerdo con las Convenciones Abiertas de Empaque.

Puede ver la estructura del archivo y los archivos simplemente cambiando el nombre de cualquier archivo DOCX a un archivo ZIP y descomprimiéndolo, como se muestra en la Figura 1 .

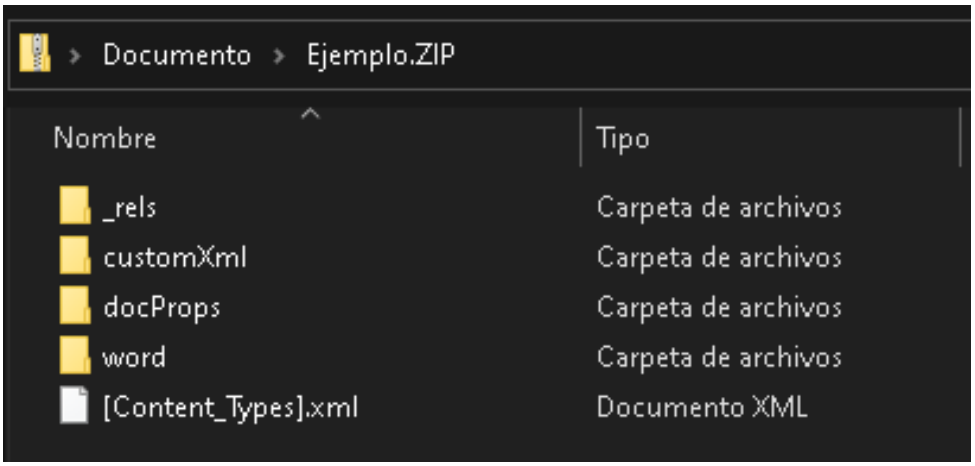


FIGURA 1: Estructura de un documento de tipo Word, visto cómo ZIP. Fuente: Autores.

Cada paquete debe tener un [Content_Types] XML, que se encuentra en la raíz del paquete. Este archivo contiene una lista de todos los tipos de contenido de las partes del paquete. Cada parte y su tipo deben aparecer en [Content_Types] XML, es importante tener esto en cuenta al agregar nuevas piezas al paquete.

Cada paquete contiene una parte de relaciones que define las relaciones entre las otras partes y los recursos fuera del paquete. Esto separa las relaciones del contenido y facilita el cambio de relaciones sin cambiar las fuentes que hacen referencia a los objetivos.

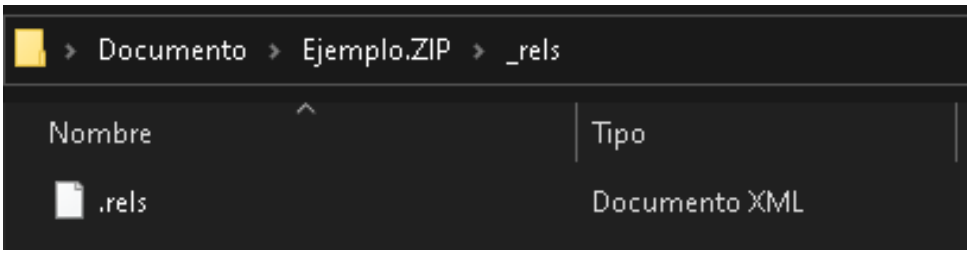


FIGURA 2: Archivo de relaciones de los recursos fuera del paquete de un documento Word. Fuente: Autores.

Para un paquete OOXML, siempre hay una parte de relaciones (.rels) dentro de la carpeta _ rels que identifica las partes iniciales del paquete o las relaciones del paquete como se muestra en la Figura 2. Por ejemplo, lo siguiente define la identidad de la parte inicial del contenido:

<Id. De relación = rId1"Type = "http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument" Target = "word / document.xml/>.

Este archivo de relaciones se puede evidenciar en la Figura 3 dentro la carpeta _rels de un archivo de DOCX descomprimido.

También suele haber relaciones dentro de .rels para app.xml y core.xml.

En este apartado de relaciones del directorio, cada parte de relaciones de origen se encontrarán dentro del directorio _rels y se configuran adicionando como extensión .rels al nombre de la sección.

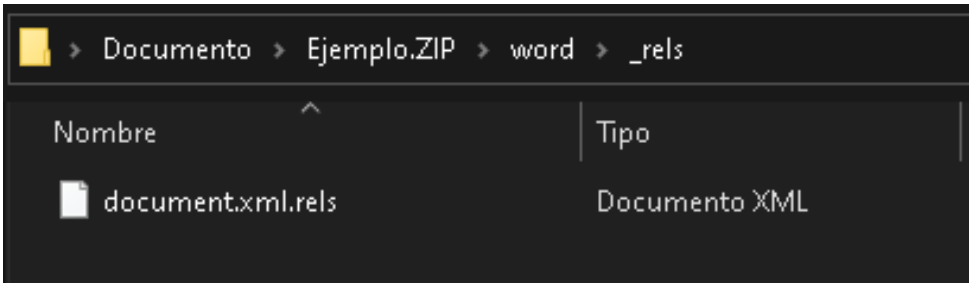


FIGURA 3: Archivo de relaciones del contenido principal de un documento Word.
Fuente: Autores.

6.2.1. Partes de los documentos WordprocessingML

La Tabla 1 muestra una lista de las posibles partes de un directorio *WordprocessingML* . Cabe aclarar que un documento solo puede tener una fracción de las secciones mencionadas.

Parte	Descripción
Comments	Contiene los comentarios del documento. Esto hace referencia los comentarios de edicion del documento.
Document Settings	Especifica la configuración del documento, tales como idioma para el corrector de errores ortográficos y gramaticales, seguimiento de revisiones, protección contra escritura, entre otros.
Endnotes	Contiene las notas al final del documento.

Font Table	Especifica la información sobre los tipos de caligrafía utilizadas en el documento. esta información sirve para determinar qué fuentes se deben utilizar para mostrar el documento, cuando la tipografía especificada no está disponible en el sistema.
Footer	Contiene la información de los pies de páginas, sus numeraciones y membretes.
Footnotes	Contiene las notas al pie del documento, tales como los comentarios o referencias inmediatas.
Glossary	Esta sección contiene una tabla dinámica de vocabulario definido por el usuario, que ayuda a la navegación dentro del mismo.
Header	Contiene la información del encabezado, numeración de la página y membretes.
Main Document	En esta sección se incluye el cuerpo del documento.
Numbering Definitions	Contiene la estructura de cada una de las numeraciones presentes en el documento.
Style Definitions	Contiene los conjunto de estilos definidos y utilizados por el documento. Esto hace referencia a las configuraciones de títulos, subtítulos, descripciones y similares.
Web Settings	Contiene las definiciones de la configuración web del documento, para el tratamiento de la información al momento de ser exportados como documentos HTML.

TABLA 1: Partes de un paquete WordprocessingML [49]

6.3. Titulación

La Titulación es el resultado del análisis que realiza el relator y su equipo de trabajo, respecto de cada providencia, concepto y decisión de definición de conflictos de competencias administrativas remitidos a la Relatoría en medio electrónico. La titulación corresponde a los descriptores y restrictores incluidos en el inicio de cada documento jurisprudencial y, una vez concluida, hace parte de la información contenida en la base de datos de información jurisprudencial de la Corporación.

1. **Descriptor:** Término utilizado para definir el contenido en un texto; este es utilizado para clasificar y almacenar las providencias.

2. **Problema Jurídico:** Se considera como una pregunta que se formula la sala de decisión a partir de los hechos del caso y del aspecto jurídico considerado en el mismo.

3. **Tesis:** Es la respuesta dada al problema jurídico explícita o implícitamente planteado en la sentencia, auto, concepto o decisión de definición de conflictos de competencias administrativas y que da solución al caso. El proceso de obtención o extracción de la tesis se hará, de ser posible, en forma literal (textual, exacta, sin parafraseo) del contenido del documento en proceso de relato. La tesis deberá guardar necesaria relación entre el supuesto de hecho y la razón de la decisión y deberá ser comprensible y expresarse con claridad. Es por esto que, la labor principal del relator y su grupo de trabajo es el análisis del documento jurisprudencial y la extracción de la tesis sustento de la decisión.

4. **Fuente formal:** Normas que sirven de sustento a las tesis obtenidas en las providencias, conceptos y decisiones de definición de conflictos de competencias administrativas.

5. **Nota de relatoría:** Es aquella precisión, alusión o comentario que efectúa el relator con el fin de referenciar o aclarar información de jurisprudencia relacionada con la tesis y citada en la providencia, concepto y decisión de definición de conflictos de competencias administrativas. También hay lugar a incluirla cuando alguna providencia se ha dejado sin efectos por decisión judicial posterior.

6.4. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural se fundamenta principalmente en la comprensión del lenguaje natural (de humano a máquina) y la generación de lenguaje natural (de máquina a humano). Para dar paso al desarrollo de metodologías basadas en estas técnicas de transformación se debe analizar y ejecutar los procesos mostrados en la Figura 4, según las características del corpus y que el objetivo requiera:

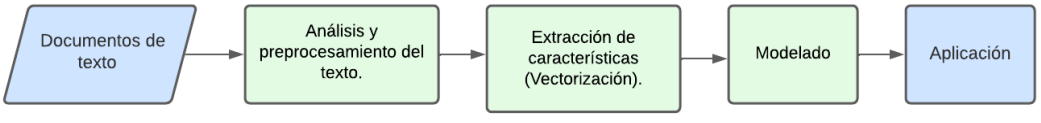


FIGURA 4: Procesamiento del Lenguaje Natural. Fuente: Autores.

6.4.1. Preprocesamiento de texto

El preprocesamiento de los datos consiste en la separación, simplificación y normalización de la información, para facilitar la comprensión de la máquina sobre los datos a procesar. Este proceso comienza con la generación de una tupla de las palabras frecuentadas en los documentos, separándolas mediante funciones de tokenización. Una vez tokenizado se realiza el proceso de limpieza de las palabras. En este proceso se excluyen de las variables los símbolos o palabras que cumplan los patrones especificados y que son irrelevantes para su análisis (abreviaciones, palabras compuestas por símbolos, palabras mal escritas, *stopwords*).

Dado el proceso de exclusión de palabras, también se puede aplicar según sea el caso en el preprocesamiento de texto, la simplificación de las palabras no excluidas por medio de los métodos de *stemming* y lematización.

El *stemming* es el proceso de convertir las palabras a su forma raíz mediante la eliminación de sufijos. Sin embargo, la palabra suele perder su significado contextual cuando se lleva a su forma raíz, como ocurre con el ejemplo de las palabras bombero y bomba (bomb-) [50].

De manera similar la lematización es el proceso de agrupar las diferentes formas flexionadas de una palabra para que puedan analizarse como un solo elemento. La lematización es similar a *stemming*, pero aporta contexto a las palabras. Por lo tanto, vincula palabras con un significado similar a una palabra. [51]

6.4.2. Balance de datos

El proceso de Balance de Datos se encarga de analizar e igualar la proporción de los datos entre las diferentes categorías. Si hay un desequilibrio, la función duplicará la información de cada categoría hasta la categoría con mayor número de documentos o reducirá la información de cada categoría hasta alcanzar la longitud de la categoría con menor número de documentos.

6.4.3. Vectorización de documentos y palabras

La vectorización es un proceso de transformación de un texto o conjunto de documentos de texto a un vector o matriz de características numéricas, mediante el mapeo de tokens. Los valores de estas representaciones vectoriales son definidos en base a las ocurrencias (frecuencias) de los tokens en los documentos, siendo modificado este valor según las reglas o fórmulas de las técnicas de vectorización (Recuento de palabras, TF-IDF, PMI) [51].

La frecuencia de las palabras puede ser indicativo de un patrón que no necesariamente significa que estas tengan correlación con alguna de las categorías que se plantean para clasificar.

6.4.4. Modelo de referencia (Baseline)

Para comenzar en el desarrollo de un algoritmo es necesario plantear un modelo de referencia. Esta clase de modelos hace referencia a un modelo básico utilizado para dar guía en la selección de los algoritmos planteados, esperando que la aplicación de estos logre demostrar una mayor exactitud que el modelo base. Esto permite dar a conocer un punto de referencia entre los clasificadores[52].

6.4.5. Modelo de N Gramas

Un modelo de N-gramas es un tipo de modelo de lenguaje probabilístico utilizado para predecir el siguiente elemento de dicha secuencia en la forma de un modelo de Márkov de orden *(n - 1)*. Dos ventajas de los modelos de N-gramas son la simplicidad y la escalabilidad; con una *n* más grande, un modelo puede almacenar más contexto con una compensación espacio-tiempo bien entendida, lo que permite que los pequeños experimentos se escalen de manera eficiente [53].

6.4.6. Modelo de TF-IDF

TF-IDF es una medida que determina el peso de una palabra en un documento en relación con otros. Por medio de este se logra facilitar procesos relacionados con la minería de texto, la búsqueda y la recuperación de información. Este modelo se llega a determinar por medio de las ecuaciones 6.1 y 6.2. La primera ecuación corresponde a la TF (*Term Frequency*) que mide la frecuencia de las palabras en cada documento y la segunda corresponde a la IDF (*Inverse Document Frequency*) que mide la rareza de la palabra en los documentos.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \tag{6.1}$$

Donde $n_{i,j}$es el recuento bruto de un término en un documento y $\sum_k n_{i,j}$ es el número de documentos en los que aparece el término n aparece.

$$IDF(\omega) = log\left(\frac{N}{df_r}\right) \tag{6.2}$$

Donde N es el número total de documentos del corpus y df_t el número de documentos que contienen la palabra ω. [54]

6.4.7. Modelo de SVM

El modelo de SVM es un conjunto de algoritmos de aprendizaje enfocado en la resolución de problemas de clasificación y regresión, desarrollado originalmente para la clasificación binaria. Este modelo se fundamenta en la construcción de un hiperplano para la agrupación y clasificación de los datos en un espacio de dimensiones n.[55]

6.4.8. Modelo de Pipeline

El modelo de *Pipeline* consiste en la aplicación secuencial de un conjunto de métodos para automatizar el flujo de trabajo de un modelo ML [56]. Dos ejemplos de los métodos que se pueden utilizar son el método de selección *K-Best* y el clasificador *Random Forest*. El método *K-best* se encarga de separar los elementos *k* con las mejores puntuaciones dentro del vector[57]. Por otro lado, el método *Random Forest Classifier* desarrolla diferentes algoritmos de árboles de decisión y termina seleccionando el resultado ideal entre los sistemas.[58]

Capítulo 7

Diseño metodológico

Dada la naturaleza del proyecto es necesario para dar paso al desarrollo del mismo una metodología centrada en lo experimental y el prototipado. El proyecto al ser en conjunto con el Consejo de Estado requería definir una metodología que actualizara de forma progresiva los requisitos especificados, para dar solución al problema. Por estas razones, se definió en intervalos irregulares de tiempo una demostración del progreso logrado en cada una de las etapas para cada *software*. Según el progreso, se sugerían cambios en cada una de las especificaciones. En la Figura 5 se puede observar la estructura correspondiente a la metodología propuesta.

7.1. Fases de la metodología

- **Planeación y refinamiento de los requisitos:** Esta fase corresponde a la fase inicial del proyecto. Dado el problema planteado, se comienza poniendo en contexto a los miembros del equipo sobre como se realizan las labores de estandarización y titulación. Para ello se debe plantear una reunión inicial con relatores de la Sección Primera del Consejo de Estado donde describan las reglas y las definiciones que deberán componer los requisitos de los programas a desarrollar. Del mismo modo si en la última etapa no se llega a un acuerdo total de los resultados, se realiza una nueva reunión donde se redefinan de los requisitos según las observaciones y consideraciones de los relatores. Una vez definidas se procederá a la fase de diseño.
- **Diseño:** Haciendo uso de los conocimientos obtenidos en el aprendizaje sobre el proceso de relatoría, el Aprendizaje automático y las diferentes técnicas de Procesamiento de

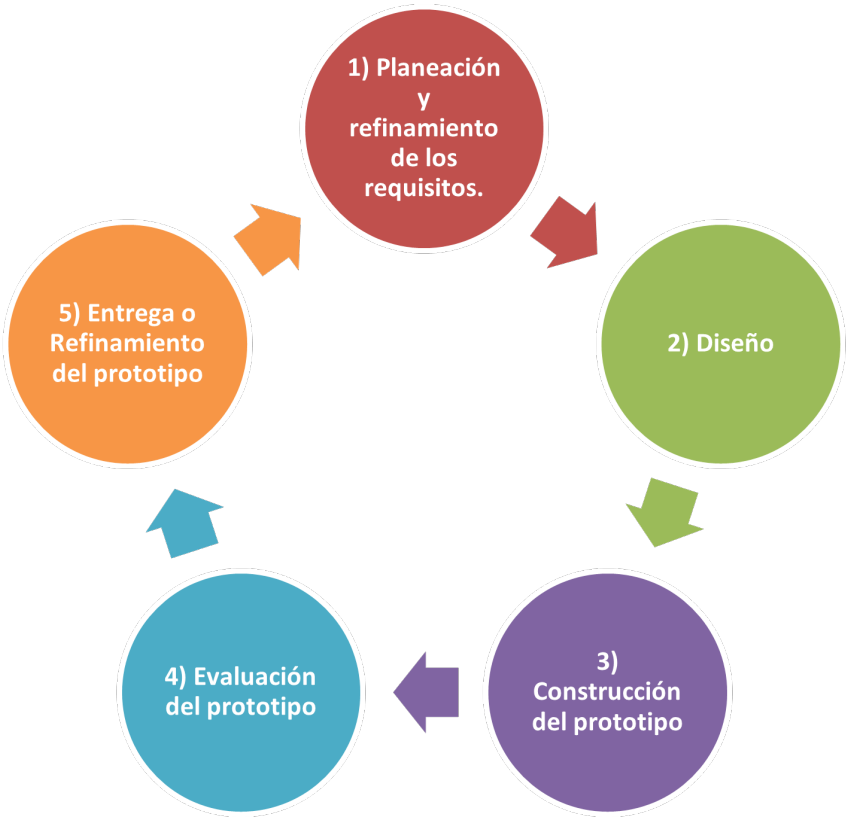


FIGURA 5: Metodología de prototipo. Fuente: Autores.

Lenguaje Natural, se analizarán y plantearán los diferentes pasos, instrumentos, datos, esquemas y funciones que darán cumplimiento a los requisitos de *software*. Una vez el diseño propuesto cumple con los requerimientos estipulados por el manual del relator, se procede a la construcción del prototipo.

- **Construcción del prototipo:** En esta fase se realizará el desarrollo de los distintos *softwares* y todo el procedimiento experimental relacionado al entrenamiento de los modelos. Una vez desarrollado un primer prototipo para cada programa se procede a la fase de evaluación.

- **Evaluación del prototipo:** En esta fase, se debe realizar nuevamente una reunión con los relatores del Consejo de Estado. Esto se realiza con el objetivo de mostrar los logros alcanzados durante la etapa de desarrollo, y así obtener una realimentación de los programas desarrollados.

- **Entrega o refinamiento del prototipo:** En esta última etapa**,** según los comentarios dados por los relatores se determina si el prototipo debe continuar con su desarrollo, ya sea

incluyendo nuevos requerimientos o arreglando problemas no planteados. De no ser el caso se da fin al proyecto realizando la entrega del producto.

7.2. Población y muestras

La población para la cual se realiza el proyecto corresponde a los relatores de la Sección Primera del Consejo de Estado, ante la labor de estandarización y de titulación de las jurisprudencias.

Un relator no llega a trabajar completamente en ambos procedimientos o al menos no por día, exclusivamente. El tiempo requerido para cada proceso puede variar en gran medida según los factores que se tengan poca claridad o las características del formato que simplemente no se cumplen. De igual manera, la experiencia y los criterios de cada relator influyen en estos tiempos. Dado lo antes mencionado para tener una mayor claridad de ambos procesos, en el desarrollo del proyecto, se debe realizar una encuesta dentro del Consejo de Estado, que estime los tiempos y los resultados de estos procesos. En adición a esta información, para dar a conocer en detalle esta labor se deberá acceder a los resultados de la labor de estandarización y titulación de la Sección Primera del Consejo de Estado. Estas jurisprudencias se encuentran almacenadas en la base de datos interna que la entidad posee.

7.3. Instrumentos

Los métodos planteados para la recolección de los datos corresponden al desarrollo de una encuesta por medio del uso de la plataforma de *Google Forms* además de la descarga de la documentación de las jurisprudencias por medio de la base de datos del Consejo de Estado.

Para dar desarrollo al proyecto, se hace uso de diferentes equipos de cómputo configurados con el sistema operativo de *Windows*. Para el desarrollo del proyecto se usaron los siguientes equipos que se describen a continuación con las siguientes características:

Equipo de cómputo 1, utilizado para el entrenamiento de los modelos y desarrollo de aplicaciones:

- Procesador AMD Ryzen 5 3600 / 6 núcleos.
- Disco duro de 1 Tb.

- Memoria RAM de 32 Gb
- Sistema operativo Windows 10

Equipo de cómputo 2, utilizado para el desarrollo de aplicaciones:

- Procesador Intel core i5 8250U / 4 núcleos
- Disco duro de 1 Tb.
- Memoria RAM de 8 Gb.
- Sistema operativo Windows 10.

Estos equipos se deben encontrar integrados con los siguientes programas para el desarrollo de los *softwares*:

- La distribución libre de Anaconda.
- El programa Qt Designer.
- El entorno de desarrollo integrado Spyder.
- La plataforma Jupyter Notebook.
- El conjunto de aplicaciones de Microsoft Office.

Dado que el proyecto se plantea realizar en *Python* los equipos deben contar con las siguientes librerías de *software*:

Para el desarrollo general de los programas:

- Numpy – Librería numérica de Python.
- difflib – Modulo de comparación de secuencias.
- python-docx – Generación, edición y lectura de documento de formato DOCX.
- re - Regular expression operations.
- collections — Tipos de datos contenedor.

- os - Interfaces misceláneas del sistema operativo.

Para el entrenamiento y aplicación de los modelos:

- random —Generar números pseudoaleatorios.
- string — Operaciones comunes de cadena de caracteres.
- pickle — Serialización de objetos Python.
- Matplotlib – Creación y visualizaciones de estáticas, animadas e interactivas en Python.
- SciKit-Learn – Librería para aprendizaje automático de software libre para Python.
- NLTK – Construcción de programas en Python para trabajar con datos del lenguaje humano.
- Spellchecker – Corrección de ortografía en Python.

Para el desarrollo de las aplicaciones:

- PyQT5 – Conjunto de librerías de Python para el marco de aplicaciones.

7.4. Recolección de datos

7.4.1. Encuesta de los tiempos de ejecución y resultados de la labor de los relatores.

Para la realización de la encuesta, se generó un formulario por medio de la plataforma de *Google Forms*. La encuesta se encuentra generada con las siguientes preguntas:

1. ¿Aproximadamente cuantas sentencias estandariza por día?
2. ¿Aproximadamente cuánto tiempo se demora estandarizando UNA (1) sentencia? Por favor indique el tiempo utilizado en Minutos.
3. ¿Aproximadamente cuantas sentencias titula por día?
4. ¿Aproximadamente cuánto tiempo se demora titulando UNA (1) sentencia? Por favor indique el tiempo utilizado en MINUTOS

Este formulario se entrega a los relatores de la Sección Primera del Consejo de Estado, para su posterior diligenciamiento. Una vez todos los miembros disponibles den repuesta al formulario, por medio de la misma plataforma se realiza la descarga de los resultados de forma gráfica.

7.4.2. Recolección de la base de datos

Para la elaboración de la Base de datos de las Titulaciones generadas por el Consejo de Estado, se puede obtener acceso a esta documentación a través de la página del Consejo de Estado en la sección de Jurisprudencia [59]. la cual da libre acceso a la búsqueda y descarga de los documentos dentro de las secciones y salas del Consejo de Estado.

La estructura dada por los documentos a descargar es definida por el Manual de la Relatoría donde la titulación de la jurisprudencia es estructurada por el relator, el cual se encargó de realizar la lectura íntegra y detallada del documento, con el objetivo de identificar la situación fáctica tratada por la Sala de Decisión o Despacho Ponente, el o los problemas(s) jurídico(s) abordado(s) y, por ende, la razón jurídica (tesis) dada a cada problema jurídico y al caso en concreto.

La estructura de la titulación de la jurisprudencia comienza con los descriptores y restrictores dados para el caso. Esta parte es escrita únicamente en mayúscula y con su correcto uso de las tildes, separando los descriptores con ”/” y los restrictores por medio de -".

Paso seguido de la titulación de cada problema jurídico se incluye la tesis que lo resuelve. La tesis es extraída por el relator de forma textual de la providencia y no debe incluir comentarios o argumentos extras.

A continuación de la tesis se incluye la fuente formal y según sea el caso y de ser pertinente una nota de relatoría, para referenciar antecedentes jurisprudenciales citados en la providencia, concepto y/o decisión de definición de conflictos de competencias administrativas, o para informar si la providencia se ha dejado sin efecto por decisión judicial, entre otros aspectos. Por último, una vez completa la sección de la titulación, se anexa el contenido de la providencia del caso. En la Figura 6 se presenta como ejemplo, una titulación realiza en la sección de primera del Consejo de Estado.

Con la estructura ya definida para las titulaciones, es posible reconocer la tesis del caso dentro del documento. Sin embargo, este proceso no era realizado por los relatores siguiendo al pie de la letra el manual del relator, ante la carencia del mismo en fechas anteriores. Por esta razón el

RECURSO DE REPOSICIÓN – Contra decisión que negó el decreto y práctica de pruebas en segunda instancia / RECURSO DE REPOSICIÓN - Es el que procede contra autos que no sean susceptibles de apelación o súplica / RECURSO ORDINARIO DE SÚPLICA - Procede contra autos de carácter apelable / AUTO QUE NIEGA EL DECRETO Y PRÁCTICA DE PRUEBAS EN SEGUNDA INSTANCIA – Es susceptible del recurso de apelación / RECURSO ORDINARIO DE SÚPLICA – Es el que procede frente al auto que denegó el decreto y práctica de pruebas en segunda instancia / RECURSO DE REPOSICIÓN – Debe ser interpretado como de súplica

En el caso sub examine, se debe determinar, por un lado, si es procedente o no el recurso de reposición interpuesto contra el auto proferido el 11 de diciembre de 2018, por medio del cual se negó el decreto de las pruebas solicitadas por la parte demandante, en segunda instancia; y, por el otro, en caso de ser procedente, si se debe confirmar, modificar o revocar la providencia recurrida. [...] Atendiendo a que, la jurisprudencia de esta Corporación ha indicado que el recurso ordinario de súplica procede en todas las instancias contra los autos interlocutorios proferidos por el ponente, que por su misma naturaleza serían apelables de haberse dictado por el inferior, de conformidad con el artículo 183 del Código Contencioso Administrativo. Atendiendo a que el auto proferido el 11 de diciembre de 2018, por medio del cual se negó el decreto de pruebas, en segunda instancia, es de naturaleza apelable en los términos del artículo 181 ejusdem; este Despacho considera que la providencia recurrida no es susceptible de ser controvertida a través del recurso de reposición, en consecuencia, el recurso se declarará improcedente. Finalmente, en aras de garantizar la prevalencia del derecho sustancial sobre el procedimental que establece el artículo 228 de la Constitución Política y atendiendo a que el recurso que procede contra la providencia proferida el 11 de diciembre de 2018 es el de súplica; este Despacho adecuará el trámite del recurso interpuesto por la parte demandante al de súplica y ordenará la remisión del expediente al despacho del Consejero que sigue en turno, para lo de su competencia.

NOTA DE RELATORÍA: Ver providencia Consejo de Estado, Sala Plena de lo Contencioso Administrativo y Sección Primera, de 7 de diciembre de 2010, Radicación 08001-23-31-000-2009-00019-02(IJ), C.P. Enrique Gil Botero; y 8 de febrero de 2018, Radicación 11001-03-24-000-2007-00323-00, C.P. Hernando Sánchez Sánchez.

FUENTE FORMAL: CONSTITUCIÓN POLÍTICA – ARTÍCULO 228 / CÓDIGO CONTENCIOSO ADMINISTRATIVO – ARTÍCULO 180 / CÓDIGO CONTENCIOSO ADMINISTRATIVO – ARTÍCULO 181

FIGURA 6: Titulación de una providencia de la Sección Primera del Consejo de Estado. Fuente: Autores.

proceso de la extracción de la tesis y el problema jurídico era elaborado a criterio de cada uno. Debido a ello, en los documentos se puede presentar en la titulación información adicional a la tesis y una descripción del problema antes de este mismo. Es entonces que para realizar la recolección de la información es necesario revisar cada una de las ideas que lo componen, analizando si realmente es una extracción literal de la providencia, comparándolo párrafo por párrafo hasta encontrar una coincidencia con gran similitud.

7.5. La estandarización de las jurisprudencias

Cuando las jurisprudencias son remitidas al Consejo de Estado, estas suelen presentar una variedad de formato y estructura dependiendo del abogado que la presente, para ello, el Consejo de Estado debe primero estandarizar el formato de la jurisprudencia entregada al despacho antes de realizar cualquier otro proceso necesario para la titulación. La estandarización y titulación de los documentos se encuentra expresada en el **Manual de Relatoría** del Consejo de Estado. El manual sirve de instructivo para el desarrollo de las funciones de la Relatoría por parte de los empleados que allí laboren dentro del quehacer diario de sus actividades. Con ese fin, ese documento enmarca los pasos que se surten dentro de los procedimientos aprobados para el proceso de divulgación jurisprudencial en el Modelo de Gestión Integral por Procesos MGIP, tales como: a) la titulación de providencias, conceptos y decisiones de definición de conflictos de competencia y b) la atención de consultas jurisprudenciales, así como las demás labores que son propias de la dependencia, tales como la elaboración y revisión del boletín de jurisprudencia y/o publicaciones similares de la Corporación, el registro de normas demandadas, entre otras.

Antes de su inclusión en la base de datos de Consulta Jurisprudencial, los documentos objeto de titulación deben ser estandarizados. Para el efecto, se aplican los criterios técnicos de tratamiento del documento en orden a su procesamiento y registro en la base de datos que se muestran en la Figura 7 y se explican a continuación:

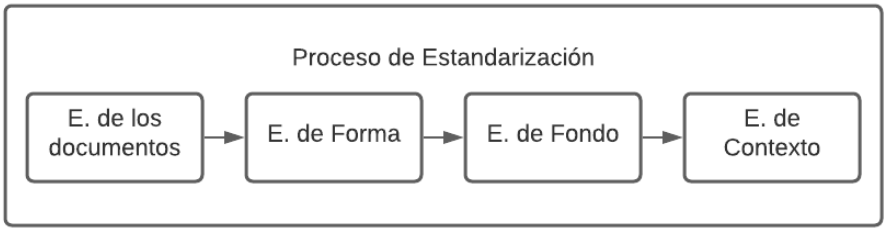


FIGURA 7: Proceso de estandarización de las jurisprudencias. Fuente: Autores.

7.5.1. Estandarización de Forma

El encabezado de las providencias, los conceptos y las decisiones de definición de conflictos de competencias administrativas debe seguir los siguientes parámetros:

Para todo el documento:

- Tamaño: Oficio (Folio 21,59 cm X 33.02 cm)
- Márgenes: Inferior 3 cm, Superior 3 cm, Derecho 3 cm, Izquierdo 3 cm
- Encuadernación 0 cm y Encabezado 0 cm.
- Encabezados: Eliminarlos
- Números de página: Eliminarlos
- Letra: Arial 12
- Formato: Documento de Word 97-2003 (*doc)

Para la titulación (Desde los descriptores hasta el demandado):

- Párrafo: sencillo
- Letra de titulación: Arial 12
- Interlineado: Sencillo (1) / Anterior (0 pto) y Posterior: (0 pto)
- Sangría: Ninguna / Izquierda (0 cm) y Derecha (0 cm)
- Fuente: Avanzado
- Escala: 100 %
- Espacio: Normal
- Posición: Normal

7.5.2. Estandarización de Fondo

Los datos generales del proceso, correspondientes al origen de la decisión, ponente, fecha de aprobación, número del expediente, demandante, demandado y asunto o referencia, según contenga o no la respectiva decisión; deben cumplir con los siguientes parámetros:

- Se debe separar con dos (1) espacios sencillos de la titulación general.
- Se inicia con la frase en mayúscula sostenida y negrilla Consejo de Estado Seguida de un (1) espacio sencillo en mayúscula sostenida y negrilla, con la frase SALA DE LO CONTENCIOSO ADMINISTRATIVO.
- Seguida de un (1) espacio sencillo en mayúscula sostenida, negrilla y con tildes, con la frase correspondiente a la sección.
- Seguida de un (1) espacio sencillo en minúscula, negrilla y con dos puntos, la frase exacta consejero ponente: o consejera ponente: según corresponda, y en mayúscula sostenida, negrilla y con tildes el nombre.
- Seguida de un (1) espacio sencillo en minúscula y con tildes, de la ciudad de origen y en letras y números, el día, mes y año de la aprobación.
- Seguida de un (1) espacio sencillo en minúscula, negrilla y con dos puntos la frase exacta Radicación número: y el número de radicación seguido del número interno entre paréntesis del proceso.
- Seguida de un (1) espacio sencillo en minúscula, negrilla y con dos puntos la frase exacta Actor: y el nombre en mayúscula sostenida y en negrita del demandante(s).
- Seguida de un (1) espacio sencillo en minúscula, negrilla y con dos puntos la frase exacta Demandado: y el nombre completo con la abreviatura correspondiente si la posee en mayúscula sostenida y negrita del demandado(s).
- Seguida de dos (2) espacios sencillos va la Referencia: o Asunto: o Tema: conforme lo diga la providencia, seguida de cuatro (4) espacios sencillos más, previos al inicio del texto de la providencia.

En los casos en que la providencia o la decisión tenga salvamentos o aclaraciones de voto, estos deberán ubicarse al final de la misma, separados por cinco (5) espacios sencillos de las firmas de los integrantes de la Sala. En estos casos se deberán atender las mismas reglas de

estandarización, con excepción de que quien profiere el salvamento o la aclaración no debe denominarse Çonsejero ponente:"sino Çonsejero:.eindicarse después de la frase alusiva a la sección.

Las aclaraciones, correcciones o adiciones de las providencias deberán ser incorporadas al final de la providencia, con los mismos parámetros previstos para la estandarización del documento, con las anteriores especificaciones el encabezado del documento deberá visualizarse como se muestra en la Figura 8.

7.5.3. Estandarización de Contexto

Una vez se haya estandarizado la providencia, concepto o decisión de definición de conflicto de competencias administrativas, el archivo deberá ser guardado en la carpeta denominada "Sentencias G.en el año y sección correspondiente, usando como nombre de archivo el número único de radicación, que consta de 23 dígitos separados por guiones seguido del interno (para las secciones segunda, tercera y cuarta) entre paréntesis del proceso.

Ejemplo:

Radicación número: 11001-03-25-000-2016-00988-00(4469-16)".

En caso de existir varias providencias del mismo proceso proferidas en el mismo año, la primera se guardará con los números único de radicación e interno entre paréntesis del proceso y las siguientes se archivarán, cronológicamente, con los mencionados números seguidos de una letra en orden alfabético.

Ejemplo:

11001-03-25-000-2016-00988-00(4469-16)

11001-03-25-000-2016-00988-00(4469-16)A

11001-03-25-000-2016-00988-00(4469-16)B

11001-03-25-000-2016-00988-00(4469-16)C

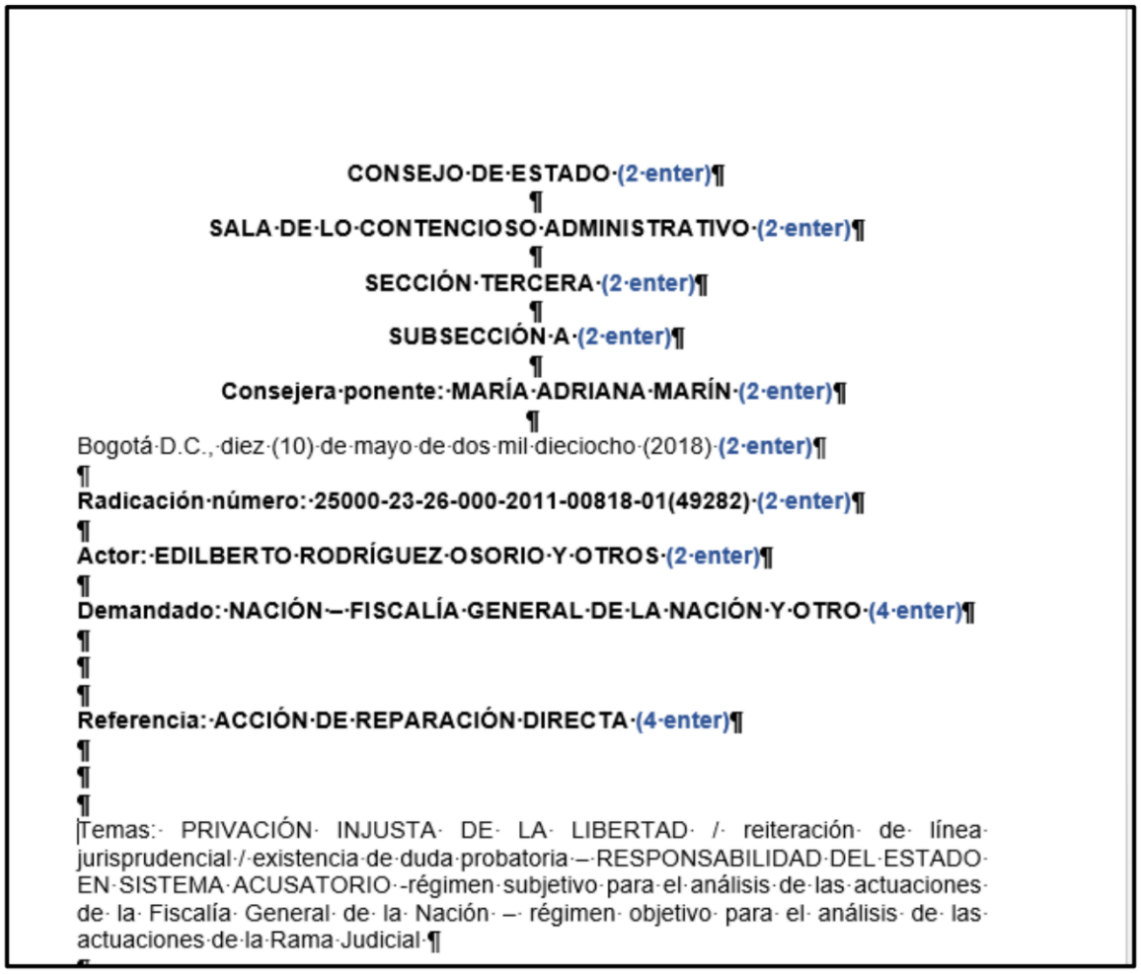


FIGURA 8: Encabezado de una jurisprudencia del Consejo de Estado. Fuente: Autores.

7.6. Requisitos de los programas

7.6.1. Requerimientos del software de estandarización de jurisprudencias

Para la automatización de los procesos descritos en la estandarización de las jurisprudencias de la Sección Primera del Consejo de Estado, se deberá realizar un ejecutable que deberá realizar las siguientes funciones. Primero se deberá separar los documentos dependiendo del formato que tienen, ya sea DOC o DOCX. Una vez se han separados los documentos en diferentes carpetas, se deberán convertir los documentos de formato DOC a DOCX, debido a que el formato DOC no es de uso libre y por lo tanto posee una escasa documentación en cuanto a estructura se refiere, a diferencia del formato DOCX que sí es de uso libre y posee

una extensa documentación sobre su estructura. Ya cuando todos los archivos se encuentran en formato DOCX se debe dar formato al documento. El formato debe ser configurado modificando las márgenes, arreglando el encabezado del documento y manteniendo un tamaño de letra fijo a lo largo todo el documento, como se indica en la metodología del proceso de estandarización del capítulo 7.5. Finalmente, como el documento se encuentra con los parámetros deseados, se debe guardar el documento con los cambios realizados.

7.6.2. Requerimientos del software para la generación de la base de datos

Dada la gran cantidad de documentos que se deberán usar y los requisitos al momento de la extracción de la información se opta por la realización de este proceso de manera automática por medio de un programa. Las funciones fijas que este programa deberá cumplir corresponderán a la lectura masiva de los documentos, la separación de la titulación de la providencia y la búsqueda de los párrafos usados para la titulación, la tesis y el problema jurídico, dentro de la providencia. Finalmente, el programa deberá almacenar esta información en una única base de datos, separado la información por documento y por clasificación, en un archivo que permita la representación de los datos en forma de tabla.

7.6.3. Requerimientos de los modelos

Para seleccionar el algoritmo idóneo para el problema objetivo, se realiza la comparación de la exactitud obtenida para cada modelo en la evaluación con los datos de entrenamiento y los datos de prueba. La primera característica que se espera es que posean un desempeño mayor al obtenido por el modelo de referencia, en ambos casos. La segunda característica que se espera para la selección del modelo es que este posea la mayor exactitud en ambos casos. Finalmente, el modelo deberá ser guardado como un archivo PKL, para su lectura en la interfaz que se desarrollará para la titulación de las jurisprudencias.

7.6.4. Requerimientos del software de titulación de las jurisprudencias

Para extraer y clasificar automáticamente los problemas y las tesis jurídicas en las providencias de la Sección Primera del Consejo de Estado, se deberá realizar un ejecutable que cumpla con las siguientes funciones: Primero el ejecutable deberá solicitar la dirección de la carpeta en la cual se encuentran los documentos a procesar, para así poder realizar la lectura de cada documento. Posteriormente deberá acceder a los modelos entrenados para extraer y clasificar

automáticamente los problemas y las tesis jurídicas en las providencias. Haciendo uso de los modelos entrenados se realiza una clasificación de cada uno de los párrafos de cada documento. Esta información deberá pasar a ser guardada en una nueva carpeta, creada en la carpeta de los documentos, donde se guarda una copia de los documentos procesados. Estas nuevas copias se les deberá modificar el principio del documento, para añadir los párrafos identificados como tesis y problema jurídico, como una combinación de los párrafos clasificados como tal.

7.7. Diseño y ejecución

7.7.1. Software de estandarización de las jurisprudencias

El primer paso de la metodología del proyecto correspondiente a los programas corresponde al desarrollo del *software* de estandarización de formato para las sentencias de la Sección Primera del Consejo de Estado. Para esto se propone un programa en el lenguaje de *Python* con la estructura propuesta en la Figura 9.

Paso 1: El programa comienza con la definición de la función *IsFolder()*, en ella se especifican y/o se generan las carpetas en las cuales se van a separar los documentos DOC y DOCX para conveniencia en ejecución del programa. Y posteriormente se llama a la función *DocumetArrangement()* donde seleccionan los documentos que se van a procesar.

Paso 2: Posteriormente se define la función *doc_docx_Converter*() donde se realiza la conversión de DOC a DOCX. El corazón de esta función radica en el uso de la librería *win32* para así hacer uso de *Microsoft Office* como conversor de archivos.

Paso 3: Una vez todos los archivos se encuentran en formato DOCX se llama la función *docxFormatter()*, que se encarga de darle formato al documento, borrando el encabezado y generando uno nuevo con la información contenida, modificando además las márgenes y el tamaño de letra a lo largo de todo el documento, según las reglas ya definidas. Teniendo el documento con los parámetros deseados, se guarda automáticamente el documento.

Paso 4: Con el código base del programa, se procede al desarrollo de una interfaz gráfica de usuario que les permita a los relatores hacer uso del programa de forma simple y accesible. Esta interfaz debe incluir las siguientes 2 funciones. La primera función llamada *SearchDOC*(), se activa por medio de la selección de un botón("...") para abrir una ventana de diálogo que permite buscar la carpeta donde se encuentran los documentos; si la dirección existe, habilita

el botón enlazado a la segunda función. La segunda función llamada *onclick*(), se activa al presionar el botón "Dar Formato". En esta función se llama a la estructura base del programa.

7.7.2. Software para la creación de la base de datos

Para la elaboración de la Base de datos de las Titulaciones generadas por el Consejo de Estado, se obtuvo la documentación por medio de una colaboración con los relatores de la Sección Primera del Consejo. En ella se realizó la recolección de 9.797 documentos en formatos DOC y DOCX, pertenecientes a los años del 2010 al 2021. Con la documentación recolectada se procede al desarrollo del programa planteado para lograr el objetivo de la extracción de la base de datos. Para ello se propone un programa en el lenguaje de *Python* con la estructura propuesta en la Figura 10.

Paso 1: El programa realizado para dar solución a la creación de la base de datos comienza con el llamado de una función definida como *docReader ()*, en la cual se ingresa como argumento la lista de documentos DOCX en la carpeta a procesar.

Paso 2: Posteriormente por medio de la librería *python-docx* se realiza la lectura de cada uno de los documentos, extrayendo únicamente los párrafos que los componen como una lista de *strings*.

Paso 3: Los documentos a procesar por el programa corresponden a documentos ya titulados, por lo cual comienza a procesar la información con una función definida como *IdTitulación()*. Esta función se encarga de la identificación del rango de párrafos que corresponden a la titulación por medio de la identificación del párrafo correspondiente a la frase inicial de la providencia (Consejo de Estado en mayúscula sostenida y negrilla), la cual sirve de indicativo como final de la titulación e inicio de la providencia.

Paso 4: Una vez extraída la titulación, por medio de una función *IdProblema()* se realiza la búsqueda del rango de párrafos correspondientes al problema jurídico. Para esto se identifica que el problema jurídico dentro de la providencia que se debe encontrar delimitada por un subtítulo numerado, con las palabras "problema jurídico" o el plural del mismo. Es por ello por lo que se realiza una búsqueda simple del título o subtítulo que contenga las palabras "problema jurídico" o que posea una similitud a estas mismas (mayor al 55%), por las pequeñas diferencias que pueda haber en los documentos, como la pluralización de las palabras, por medio de la librería *difflib.* Esta librería, contiene un método llamado *SequenceMatcher* que entrega un valor de 0 a 1, siendo 1 la coincidencia exacta entre dos cadenas de texto, por medio del algoritmo *"gestalt pattern matching"*. Con el subtítulo

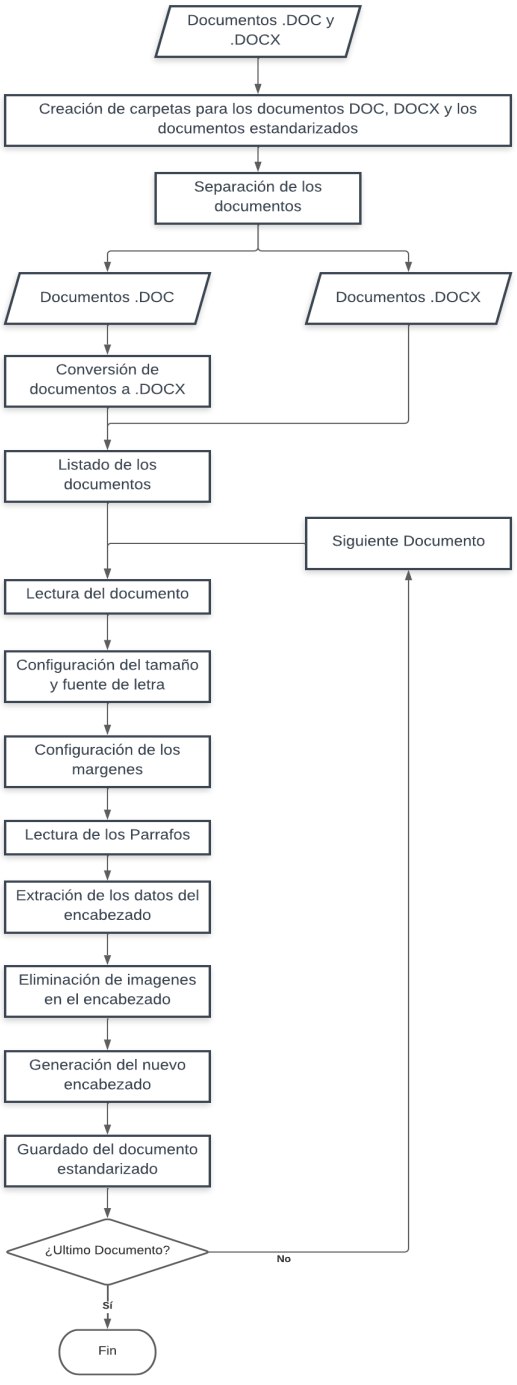


FIGURA 9: Proceso de Estandarización de las jurisprudencias. Fuente: Autores.

encontrado se determina el rango de párrafos desde este punto, hasta el próximo título o subtítulo presente en el documento.

Paso 5: Se realiza la búsqueda de la Tesis por medio de una función definida como *IdTesis().* Dentro de la providencia no llega a haber un indicativo claro de los párrafos correspondientes a la tesis, por lo cual, se realiza la búsqueda haciendo uso de la titulación, aprovechando la tesis asignada para el documento. Debido a que el párrafo de la titulación corresponde a la unión de los párrafos del documento que componen la tesis, se comienza separando cada una de las ideas del párrafo, eliminando las ideas que puedan corresponder al problema jurídico. Con las ideas restantes se procede a buscar en la providencia los párrafos que lleguen a tener una alta similitud (mayor al 75 %) con cada una de las ideas. Una vez identificados los párrafos se guarda una lista con sus posiciones en el documento y se extraen cada uno de ellos.

Paso 6: Ya teniendo identificados los párrafos correspondientes a la tesis y el problema jurídico en el documento se llama a una función denominada *SaveData_CSV*(). En ella se ingresa la información del documento procesado y de los párrafos extraídos. Con esta información se procede a generar una copia de los párrafos del documento excluyendo por medio de la lista de posiciones, los párrafos correspondientes a la tesis y al problema jurídico, denominando estos párrafos como la información "Extra", necesaria para la elaboración del algoritmo. Posteriormente para los 3 grupos de párrafos que se van a guardar en la base de datos, se realiza una eliminación de los puntos (".") presentes dentro de cada párrafo, excluyendo el punto final de cada uno, y se unen en una única cadena de texto para cada grupo. Esta información pasa a ser almacenada en un archivo que permita representar los datos en forma de tablas, para componer la base de datos. Sin embargo, la longitud de los textos a almacenar en cada una de las celdas supera los 32.767 caracteres, que imposibilitaba el guardado de la información en archivos .XLS y similares. Es por esto que se almacenó la información en un archivo de formato CSV. Finalmente, en la función se realiza una lectura de la base de datos generada y se añade o reemplaza un registro con la siguiente información, según el documento procesado:

- Nombre del documento.
- Carpeta del documento.
- Problema Jurídico.
- Tesis.
- Extra

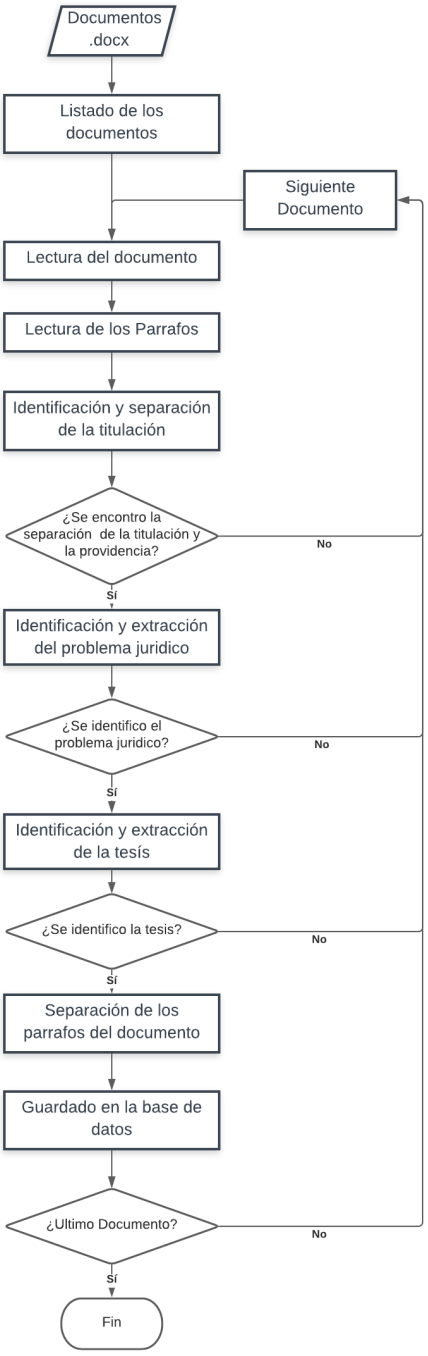


FIGURA 10: Proceso de identificación y extracción de las tesis y los problemas jurídicos. Fuente: Autores.

Una vez procesados los 9.797 documentos por medio del programa, se obtuvo una base de datos con la información de 2.167 documentos, los cuales efectivamente cumplían con los requisitos de tener en el cuerpo de cada documento, su respectiva tesis y el problema jurídico de la sentencia. La base de datos obtenida llega a representar tan solo el 22,12 % de las sentencias de la Sección Primera del Consejo de Estado entre los años del 2010 al 2021.

7.7.3. Modelado de los algoritmos

Para el desarrollo de los algoritmos se planteó la siguiente metodología que se muestra en la Figura 11. Para ello se comienza realizando una lectura de la base de datos realizada en formato CSV. Como cada registro se guarda por documento, se debe realizar una función para separar en un *dataframe* las 3 clases de información presente en los documentos: tesis, problema jurídico y extra.

Realizada la separación de la base de datos, se obtuvo como se muestra en la Tabla 2 la distribución de datos en las tres categorías.

Categoría	Tesis	Problema Jurídico	Extra
Datos	11035	16241	591245

TABLA 2: Distribución de los datos por clasificación

Con el análisis de distribución de los textos, es posible observar el desequilibrio en los datos recogidos. Sin embargo, éstos se ven limitados, ante las proporciones que llegan a representar los párrafos de tesis y de problema jurídico dentro de los documentos, a comparación de la cantidad de párrafos que no corresponden a estas clases. Por lo tanto, es necesario duplicar los datos de las minorías tantas veces como sea necesario para equiparar la cantidad de datos entre las clases y garantizar una evaluación adecuada del sistema. Como la información de cada documento se encuentra estrechamente relacionada, se opta por una repetición de los datos de tesis y de problemas hasta la cantidad de los párrafos Extra de los documentos.

Una vez terminado de organizar los datos se debe proceder al preprocesamiento de los mismos. Este paso se realiza para simplificar y reducir la información a procesar. Dado que el problema planteado de la clasificación de la información dentro las titulaciones, no se encuentra relacionado a ningún patrón de caracteres, numeración, símbolo o palabras comúnmente usadas para la articulación de textos dentro del lenguaje español, se realiza la eliminación de estos. Primeramente, todas las tabulaciones en el texto son eliminadas. Segundo toda puntuación y símbolos que no correspondan al código *ASCII* son eliminados y

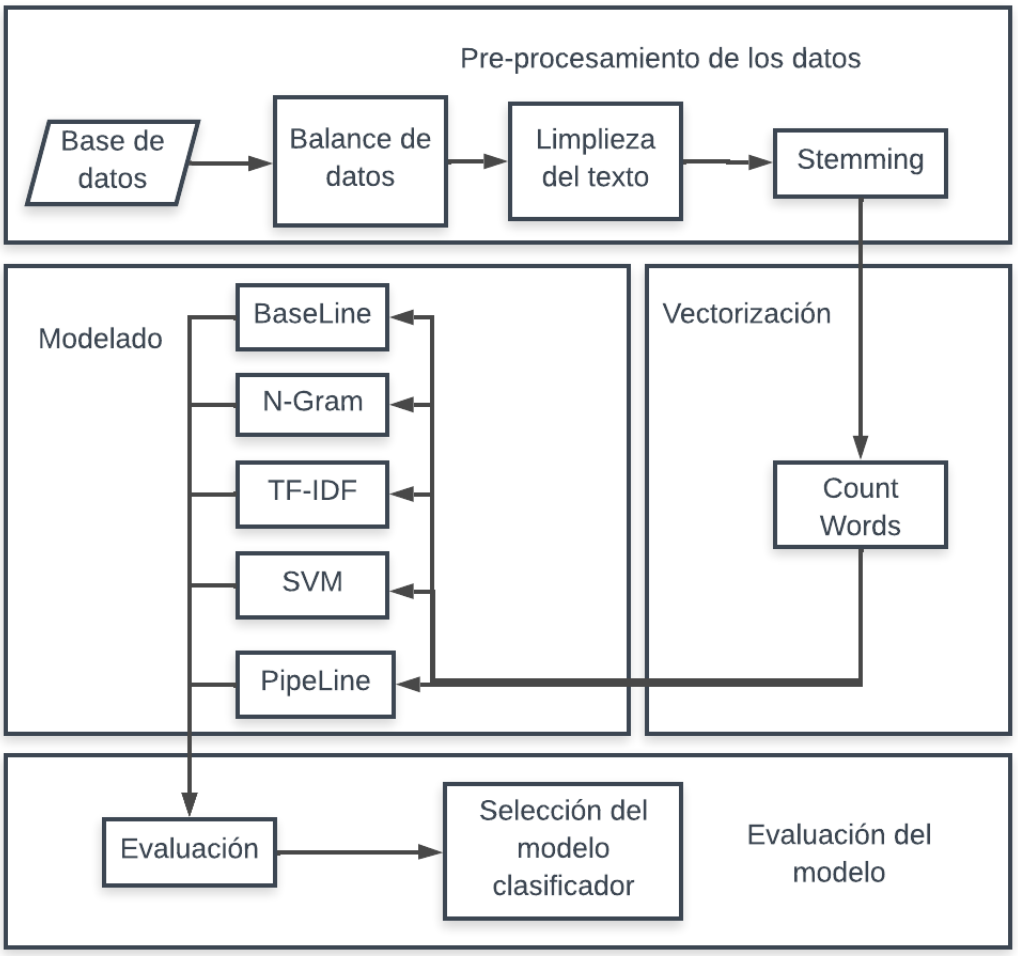


FIGURA 11: Metodología propuesta de PLN. Fuente: Autores.

reemplazados por espacios en blanco. Tercero todas las numeraciones en los párrafos son eliminadas. Cuarto toda repetición de espaciados en los textos son reemplazados por un único espacio. Finalmente se realiza una limpieza del texto eliminando las palabras con una longitud menor de 3 caracteres, realizando un filtrado de las palabras de stop *words* y realizando una reducción de las palabras por el método de *stemming*. Este proceso se aplica a cada uno de los documentos del Corpus y posteriormente se separan en las características y en las etiquetas que servirán para el entrenamiento del modelo.

Para el desarrollo del algoritmo se procede a realizar 5 distintos modelos por técnicas de Aprendizaje automático. Para el entrenamiento de los modelos se comienza realizando una

separación del conjunto de datos con una proporción del 75 % para los datos de entrenamiento. El primer modelo realizado corresponde al de *Baseline* que servirá como referencia en la evaluación de los algoritmos. El procesamiento del texto comienza con la transformación de los textos por medio de la vectorización. Este proceso se realiza para este caso separando las palabras en *unigramas*, descartando aquellos que tengan una frecuencia menor a 2, en los documentos. Paso seguido se usa el método de regresión logística y se entrena el modelo con los datos separados anteriormente. Al final del entrenamiento del modelo, se realiza una verificación de los resultados calculando las métricas de los datos de entrenamiento. Paso seguido se muestra de manera visual la clasificación de los textos por medio de una matriz de confusión. Este proceso se repite con los datos separados para la prueba del modelo, para observar el comportamiento del modelo ante los casos de entrenamiento y ante casos desconocidos para este mismo. Este proceso se repetirá para la evaluación de los siguientes modelos. El siguiente modelo utilizado corresponde a un modelo de N gramas. En este modelo se realiza la vectorización de los datos en unigramas y bigramas. Posteriormente se repite el procedimiento haciendo uso del método de regresión logística. Para el tercer modelo se repite la misma metodología, pero cambiando la vectorización de los datos, por el método de TF-IDF. Para el cuarto modelo se utiliza el método de *Support Vector Machine* para el entrenamiento del modelo. Finalmente, en el quinto modelo se realiza una modificación de la estructura del entrenamiento utilizando un modelo de *Pipeline*. En él se realiza en 3 pasos realizados secuencialmente el entrenamiento del modelo. Primero se realiza la vectorización usando los N gramas, segundo se usa el método de *select K best* y posteriormente se entrena un modelo por el clasificador de *Random Forest*. El entrenamiento de los modelos se realiza de manera repetida, hasta lograr el valor más cercano a 1 para cada caso en las métricas de clasificación.

7.7.4. Software para la titulación de las jurisprudencias

Para el desarrollo de este programa se tomó como base el código del *software* utilizado para la creación de la base de datos del Anexo 2, el código utilizado para el entrenamiento de los modelos del Anexo 3 y la interfaz de usuario generada con el código del Anexo 4. Los procesos que se espera realice el programa de titulación se presentan en la Figura 12 y se describen a continuación:

Paso 1: El programa comienza con la búsqueda del modelo entrenado, guardado como un PKL en conjunto a los datos de vectorización dentro de la carpeta de la aplicación.

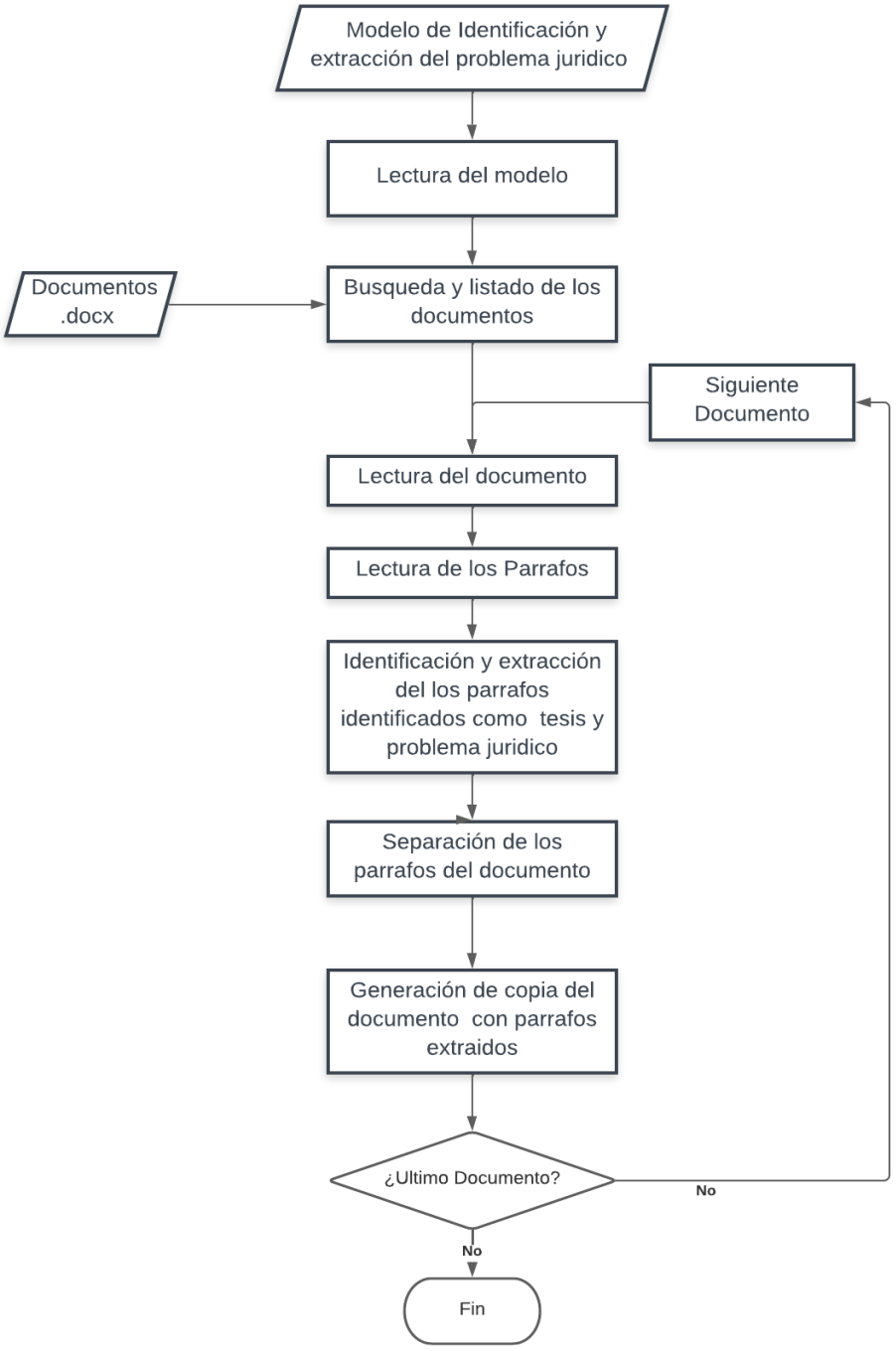


FIGURA 12: Proceso de titulación por medio del Software desarrollado. Fuente: Autores.

Paso 2: Teniendo como base el código del *software* utilizado para la creación de la base de datos, se realiza la lectura de los documentos y su fragmentación en párrafos, como una lista de cadenas de texto.

Paso 3: Se realiza el pre-procesamiento de los párrafos del documento con la función planteada en la sección 7.7.3.

Paso 4: Una vez ya los párrafos se encuentren preprocesados, se realiza la clasificación de cada una de las cadenas de texto, almacenado los resultados dentro de un vector.

Paso 5: Con los resultados obtenidos se generan dos listas donde se guardan las posiciones de los párrafos que son clasificados por el programa como tesis y problema jurídico.

Paso 6: Finalmente, se generan dos párrafos donde se combinan de manera ordenada los párrafos identificados como tesis y los párrafos identificados como problemas jurídicos, anteponiendo cada párrafo los títulos de "Tesis:" y "Problema jurídico". Estos párrafos pasan a ser copiados al principio de los documentos y guardados dentro de carpeta asignada.

Capítulo 8

Resultados

8.1. Resultados de la Estandarización de las Jurisprudencias

8.1.1. Estandarización de las jurisprudencias

Como resultado del desarrollo de un *software* de estandarización de las jurisprudencias de la Sección Primera del Consejo de Estado, se obtuvo el programa descrito en el código que se encuentra en el Anexo 1, junto con la Interfaz de usuario que se muestra en la Figura 13.

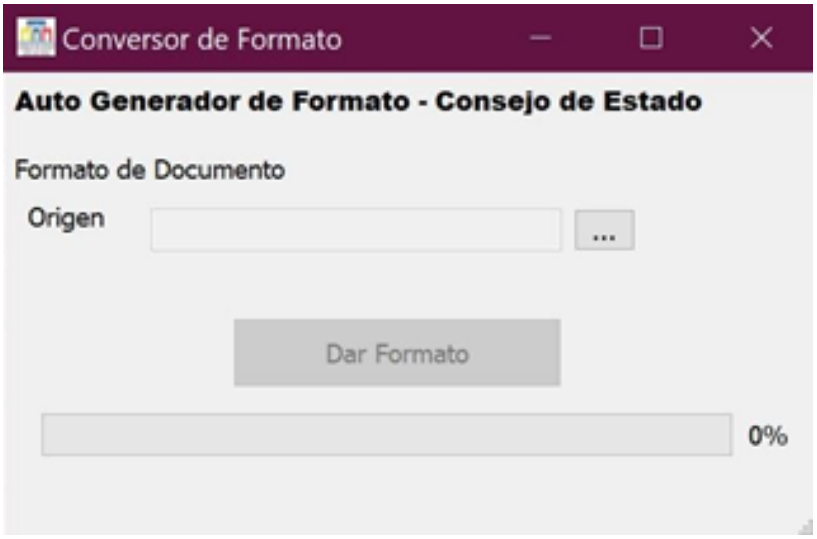
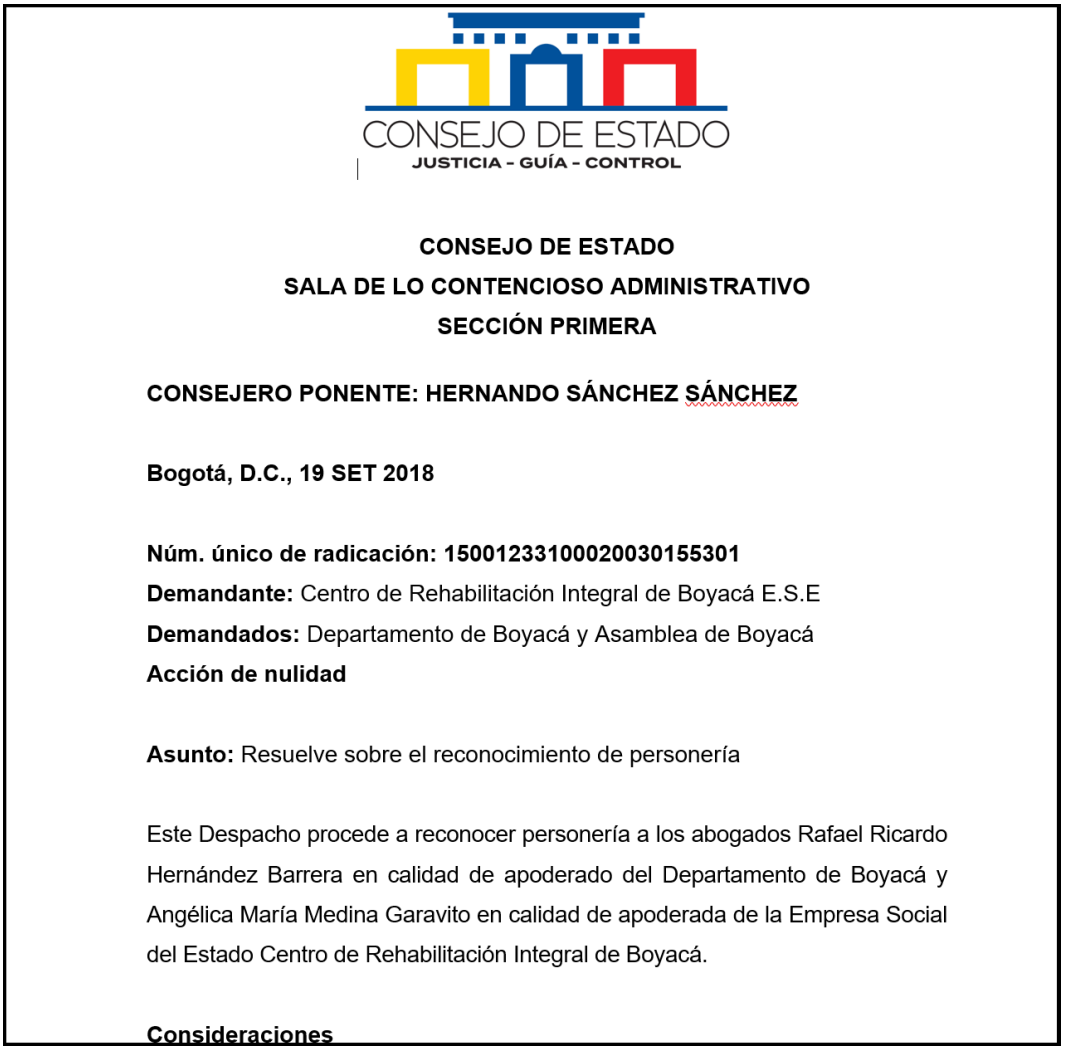


FIGURA 13: GUI para la estandarización de sentencias previo a su titulación.
Fuente: Autores.

Al aplicar el programa en una sentencia que no cumple con el formato de estandarización como el que se muestra en la Figura 14, se obtiene el documento que se muestran en la Figura 15. Como se observa en ambas figuras se realiza correctamente el proceso de estandarización. Al realizar la transformación del documento se eliminan componentes que no corresponden al encabezado como viene a ser la imagen del Consejo de Estado que se muestra al principio del documento. Siguiendo con el encabezado se corrige el formato que mostraba el documento para la fecha y el número de radicado. De igual manera se realizan los cambios de palabras que correspondían al actor y al demandado. Finalmente, en el aspecto general del documento se puede observar la modificación en los márgenes al establecido, además del cambio de tamaño y tipo de la fuente.


CONSEJO DE ESTADO
JUSTICIA - GUÍA - CONTROL

CONSEJO DE ESTADO
SALA DE LO CONTENCIOSO ADMINISTRATIVO
SECCIÓN PRIMERA

CONSEJERO PONENTE: HERNANDO SÁNCHEZ SÁNCHEZ

Bogotá, D.C., 19 SET 2018

Núm. único de radicación: 15001233100020030155301
Demandante: Centro de Rehabilitación Integral de Boyacá E.S.E
Demandados: Departamento de Boyacá y Asamblea de Boyacá
Acción de nulidad

Asunto: Resuelve sobre el reconocimiento de personería

Este Despacho procede a reconocer personería a los abogados Rafael Ricardo Hernández Barrera en calidad de apoderado del Departamento de Boyacá y Angélica María Medina Garavito en calidad de apoderada de la Empresa Social del Estado Centro de Rehabilitación Integral de Boyacá.

Consideraciones

FIGURA 14: Encabezado de un documento sin estandarizar. Fuente: Autores.

CONSEJO DE ESTADO

SALA DE LO CONTENCIOSO ADMINISTRATIVO

SECCIÓN PRIMERA

SUBSECCIÓN A

Consejero ponente: HERNANDO SÁNCHEZ SÁNCHEZ

Bogotá, D.C., diecinueve (19) de septiembre de dos mil dieciocho (2018)

Radicación número:15001-23-31-000-2003-01553-01

Actor: CENTRO DE REHABILITACIÓN INTEGRAL DE BOYACÁ E.S.E

Demandado: DEPARTAMENTO DE BOYACÁ Y ASAMBLEA DE BOYACÁ

Referencia:

Este Despacho procede a reconocer personería a los abogados Rafael Ricardo Hernández Barrera en calidad de apoderado del Departamento de Boyacá y Angélica María Medina Garavito en calidad de apoderada de la Empresa Social del Estado Centro de Rehabilitación Integral de Boyacá.

Consideraciones

1. Vistos los artículos 74 y 75 de la Ley 1564 de 12 de julio de 2012, sobre los poderes y la designación y sustitución de apoderados.
2. Atendiendo a que la abogada Angélica María Medina Garavito, identificada con la cédula de ciudadanía núm. 1.049.631.431 y con tarjeta profesional de abogado núm. 259.974, expedida por el Consejo Superior de la Judicatura, allegó poder para actuar en calidad de apoderada especial de la Empresa Social del Estado Centro de Rehabilitación Integral de Boyacá, y considerando que el poder cumple con los requisitos de ley, este Despacho le reconocerá la respectiva personería.

3. Atendiendo a que el abogado Rafael Ricardo Hernández Barrera, identificado con la cédula de ciudadanía núm. 74.382.137 y con tarjeta profesional de abogado núm. 180.354, expedida por el Consejo Superior de la Judicatura, allegó poder para actuar en calidad de apoderado especial del Departamento de Boyacá y considerando que el poder cumple con los requisitos de ley, este Despacho le reconocerá la respectiva personería.

En mérito de lo expuesto, el Despacho,

RESUELVE:

PRIMERO: Reconocer personería a la abogada Angélica María Medina Garavito identificada con la cédula de ciudadanía núm. 1.049.631.431 y con tarjeta profesional de abogado núm. 259.974, expedida por el Consejo Superior de la

FIGURA 15: Encabezado de un documento procesado. Fuente: Autores.

Posteriormente a la comprobación del programa, se realizó la medición del tiempo de demora en la estandarización de 20 documentos distintos de la Sección Primera del Consejo de Estado, obteniendo como resultado un tiempo total de 7.185 segundos que equivale a aproximadamente 360 milisegundos por documento.

8.2. Identificación de problemas jurídicos y tesis para la titulación.

El proceso de modelamiento se realizó como se indicó en la metodología del diagrama de la Figura 11. Al realizar el experimento para cada modelo planteado se obtuvieron los resultados mostrados a continuación para cada uno.

8.2.1. Entrenamiento y pruebas del modelo de referencia (Baseline).

El primer experimento realizado corresponde al entrenamiento del algoritmo con la tokenización de *unigramas* y su vectorización. Al ser el método más básico, se usa como la línea de referencia para la evaluación de las otras técnicas planteadas. En la Figura 16 se puede observar la matriz de confusión resultante al comprobar la predicción con los datos de entrenamiento del algoritmo, para el modelo de referencia.

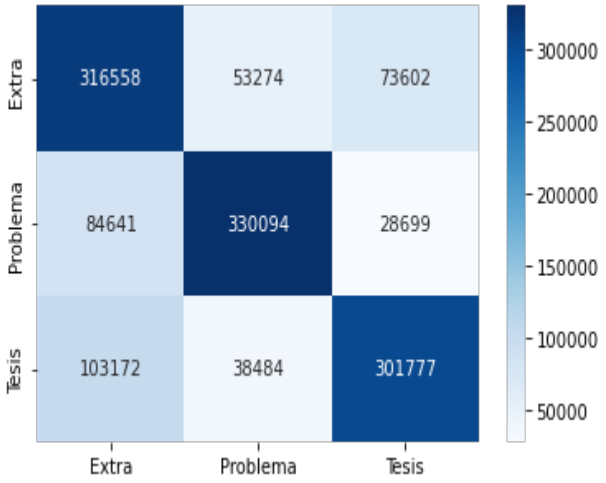


FIGURA 16: Matriz de confusión del entrenamiento del modelo de referencia.
Fuente: Autores.

En conjunto a esta información, en la Tabla 3 se puede observar la precisión, el *recall* y la exactitud, para cada categoría en la predicción con los datos de entrenamiento.

Clase	Precisión	Recall	F1 Score
Extra	0.6276	0.7139	0.6680
Problema	0.7825	0.7444	0.7630
Tesis	0.7468	0.6805	0.7121
Exactitud	0.7129		

TABLA 3: Tabla de métricas de clasificación del entrenamiento del modelo de referencia.

Una vez comprobado el correcto funcionamiento del algoritmo con los datos de entrenamiento se realiza la prueba con los datos de evaluación. El resultado de la prueba del algoritmo se puede observar en la Figura 17.

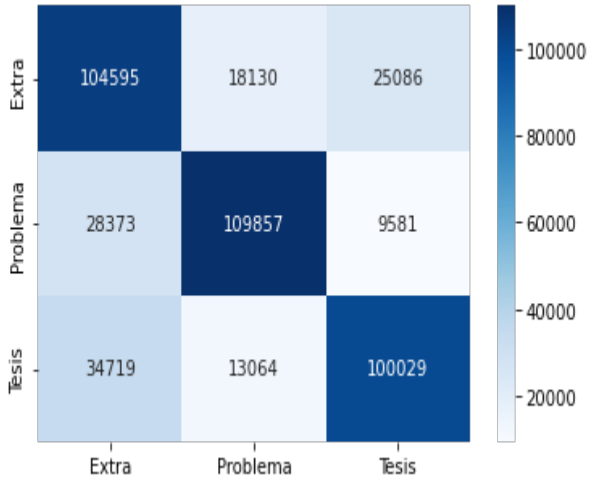


FIGURA 17: Matriz de confusión del test del modelo de referencia. Fuente: Autores.

De manera similar en la Tabla 4 se puede observar la precisión, el *recall* y la exactitud, para cada categoría en la predicción con los datos de prueba.

Clase	Precisión	Recall	F1 Score
Extra	0.6238	0.7076	0.6630
Problema	0.7788	0.7432	0.7606
Tesis	0.7426	0.6767	0.7081
Exactitud	0.7092		

TABLA 4: Tabla de métricas de clasificación del test del modelo de referencia

8.2.2. Entrenamiento y pruebas del modelo de N-Gramas.

Para el segundo experimento se desarrolló el modelo de N-Gramas. En este caso se realizó el entrenamiento teniendo como las características del modelo, los *unigramas* y *bigramas* presentes en el corpus. Dada estas propiedades se realizan los mismos pasos planteados en el modelo de referencia. En la Figura 18 se puede ver la matriz de confusión resultado del entrenamiento para el modelo de N-Gramas.

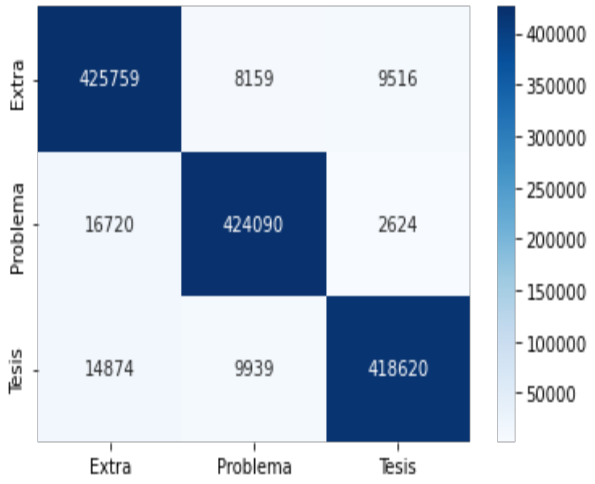


FIGURA 18: Matriz de confusión del entrenamiento del modelo de N gramas.
Fuente: Autores.

De igual forma en la Tabla 5 se puede observar las métricas de la clasificación de los datos de entrenamiento para el modelo de N-Gram.

Clase	Precisión	Recall	F1 Score
Extra	0.9309	0.9601	0.9453
Problema	0.9591	0.9564	0.9577
Tesis	0.9718	0.9440	0.9577
Exactitud	0.9535		

TABLA 5: Tabla de métricas de clasificación del entrenamiento del modelo de N gramas.

Con la comprobación del algoritmo con los datos de entrenamiento, se procede a la aplicación con los datos de prueba, obteniendo la Figura 19 correspondiente a la predicción.

Y la Tabla 6 con las métricas de la clasificación de los datos de prueba para el modelo de N-Gramas.

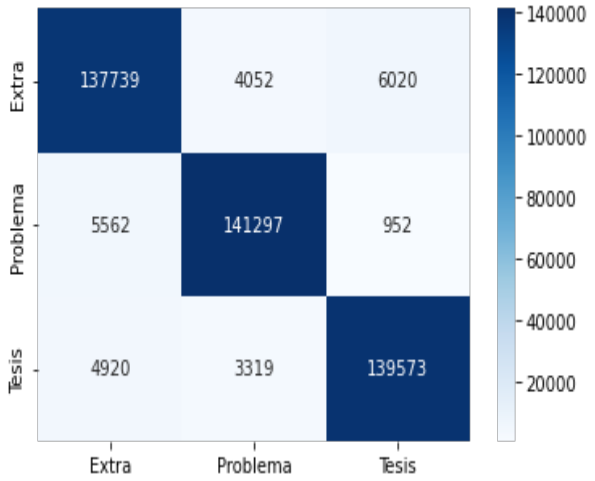


FIGURA 19: Matriz de confusión del test del modelo de N gramas. Fuente: Autores.

Clase	Precisión	Recall	F1 Score
Extra	0.9293	0.9319	0.9306
Problema	0.9504	0.9559	0.9532
Tesis	0.9524	0.9443	0.9483
Exactitud	0.9440		

TABLA 6: Tabla de métricas de clasificación del test del modelo de N gramas.

8.2.3. Entrenamiento y pruebas del modelo de TF-IDF.

En el tercer experimento, se desarrolló el modelo de TF-IDF. En este caso es necesario generar el vector siguiendo las ecuaciones (1 y(2 para el cálculo del TF y del IDF de cada uno de los unigramas. Con los resultados de estas fórmulas se retorna la respectiva multiplicación de cada uno de los valores. Con la vectorización de los datos con su valor correspondiente del método de TF-IDF, se entrenó el sistema obteniendo la matriz de confusión de la Figura 20 y las métricas de la clasificación en la Tabla 7.

Clase	Precisión	Recall	F1 Score
Extra	0.6615	0.6585	0.6600
Problema	0.7645	0.7602	0.7624
Tesis	0.7090	0.7164	0.7127
Exactitud	0.7117		

TABLA 7: Tabla de métricas de clasificación del entrenamiento del modelo de TF-IDF.

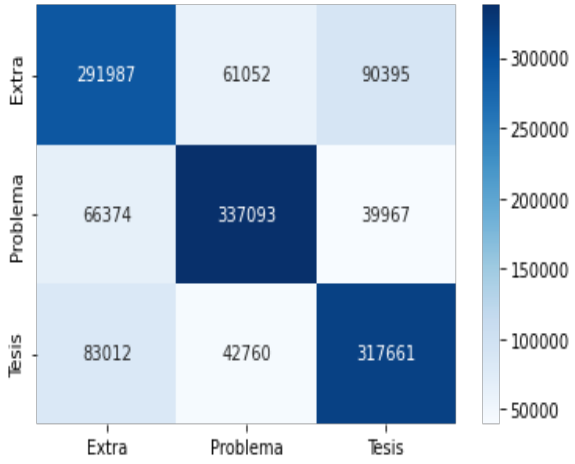


FIGURA 20: Matriz de confusión del entrenamiento del modelo de TF- IDF. Fuente: Autores.

Con la comprobación del algoritmo con los datos de entrenamiento, se evalúa con los datos de prueba para el modelo del TF-IDF. En la Figura 21 se puede observar la matriz de confusión y las métricas de la clasificación en la Tabla 8.

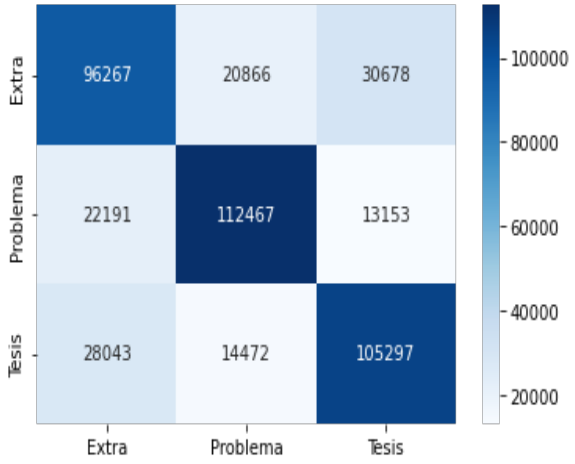


FIGURA 21: Matriz de confusión del test del modelo de TF- IDF. Fuente: Autores.

Clase	Precisión	Recall	F1 Score
Extra	0.6571	0.6513	0.6542
Problema	0.7609	0.7609	0.7609
Tesis	0.7061	0.7124	0.7092
Exactitud	0.7082		

TABLA 8: Tabla de métricas de clasificación del test del modelo de TF-IDF.

8.2.4. Entrenamiento y pruebas del modelo de SVM.

En el cuarto caso, usando el método se SVM, se realiza también la vectorización de los unigramas y bigramas. Con el vector ya definido realizamos el entrenamiento del modelo y la predicción para los datos correspondientes, como se puede observar en la Figura 22.

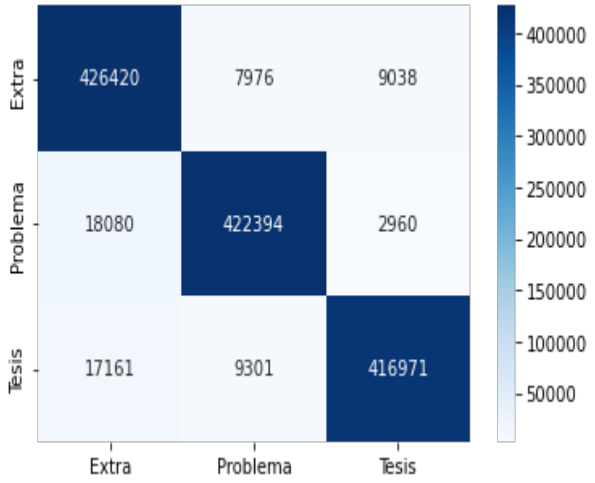


FIGURA 22: Matriz de confusión del entrenamiento del modelo de SVM. Fuente: Autores.

También se obtienen las métricas de la clasificación presente en la Tabla 9.

Clase	Precisión	Recall	F1 Score
Extra	0.9237	0.9616	0.9423
Problema	0.9607	0.9526	0.9566
Tesis	0.9720	0.9403	0.9559
Exactitud	0.9515		

TABLA 9: Tabla de métricas de clasificación del entrenamiento del modelo de SVM.

Para este modelo, al realizar la predicción para los datos de prueba se obtuvo la matriz mostrada en la Figura 23 y las métricas de la clasificación en la Tabla 10.

Clase	Precisión	Recall	F1 Score
Extra	0.9219	0.9374	0.9296
Problema	0.9533	0.9522	0.9528
Tesis	0.9555	0.9406	0.9480
Exactitud	0.9434		

TABLA 10: Tabla de métricas de clasificación del test del modelo de SVM.

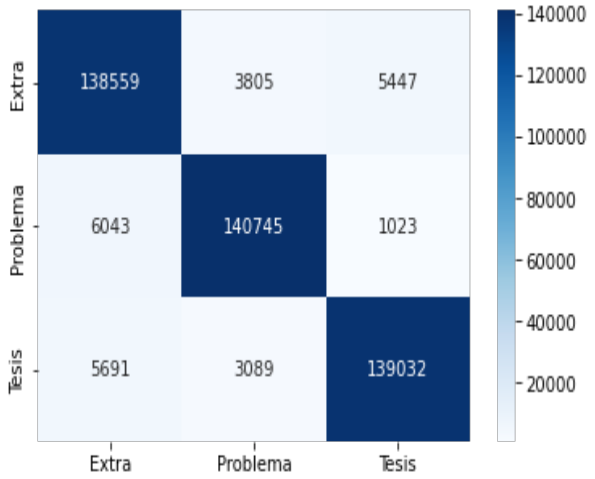


FIGURA 23: Matriz de confusión del test del modelo de SVM. Fuente: Autores.

8.2.5. Entrenamiento y pruebas del modelo de Pipeline

En el último experimento se desarrolló el modelo de *Pipeline*. El cual desarrolla de manera secuencial para los datos 3 procesos: la vectorización, la selección de los *K-Best* y la estimación del modelo por medio de *Random Forest Classifier*. En la selección de los K mejores se separan las mejores 1200 características obtenidas. Una vez entrenado el algoritmo se obtiene la siguiente matriz de confusión para los datos de entrenamiento, como se ve en la Figura 24 y la Tabla 11 con las métricas obtenidas del entrenamiento.

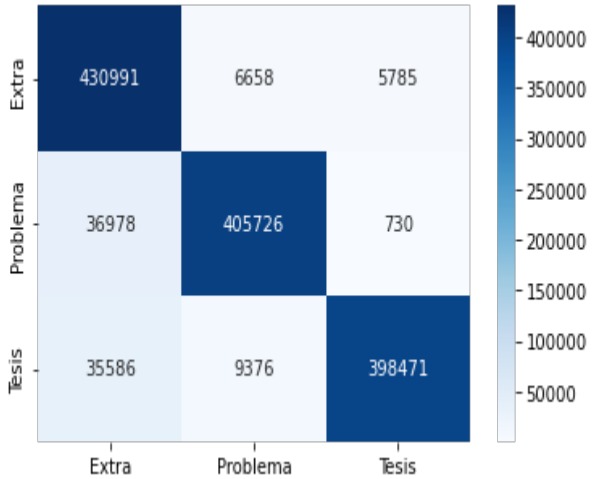


FIGURA 24: Matriz de confusión del entrenamiento del modelo de Pipeline. Fuente: Autores.

Clase	Precisión	Recall	F1 Score
Extra	0.8559	0.9719	0.9102
Problema	0.9620	0.9150	0.9379
Tesis	0.9839	0.8986	0.9393
Exactitud	0.9285		

TABLA 11: Tabla de métricas de clasificación del entrenamiento del modelo de Pipeline.

Finalmente, al realizar la predicción para los datos de prueba se obtuvo lo observado en la Figura 25 y el respectivo reporte de la clasificación en la Tabla 12.

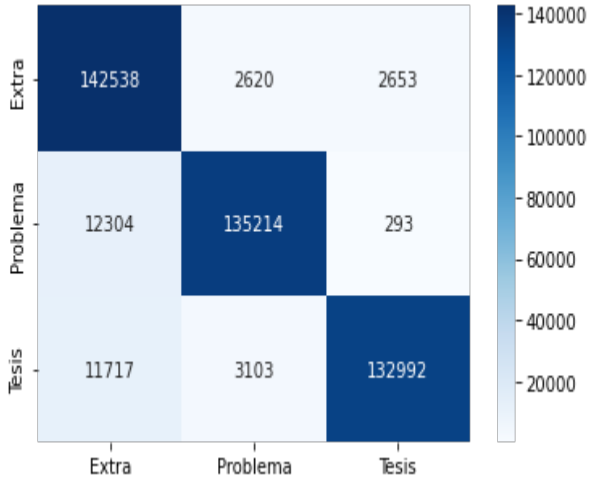


FIGURA 25: Matriz de confusión del test del modelo de Pipeline. Fuente: Autores.

Clase	Precisión	Recall	F1 Score
Extra	0.8558	0.9643	0.9068
Problema	0.9594	0.9148	0.9366
Tesis	0.9783	0.8997	0.9374
Exactitud	0.9263		

TABLA 12: Tabla de métricas de clasificación del test del modelo de Pipeline.

8.2.6. Titulación de las jurisprudencias

El programa desarrollado para el proceso de titulación de las jurisprudencias se realizó con el código del Anexo 4 generando de manera similar la Interfaz de usuario que se muestra en la Figura 26.

Al aplicar el programa en una sentencia sin titular como la que se muestra en la Figura 27 se obtuvo como resultado el documento que se muestra en la Figura 28. En esta, el programa extrae secciones de texto y las organiza de acuerdo a su clasificación (Problema jurídico o Tesis). Este texto extraído y acomodado, no es modificado de ninguna manera, tal como se define en el manual del relator, la titulación de una sentencia judicial no debe presentar modificaciones de ningún tipo, por ejemplo, errores ortográficos, palabras mal escritas o modismos presentes en la sentencia original. De manera continua, el programa extrae y organiza los diferentes extractos de manera ordenada siguiendo el orden en el que fueron escritos en la sentencia, omitiendo los conectores o párrafos innecesarios mediante el uso de cajas con puntos suspensivos los cuales indican texto adicional omitido entre cada párrafo extraído.

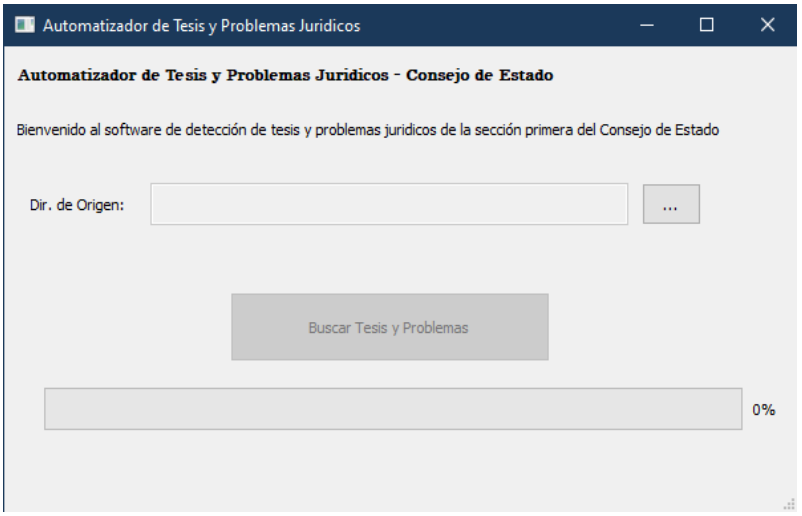


FIGURA 26: GUI para la identificación de las tesis y los problemas jurídicos de las sentencias. Fuente: Autores.

Posteriormente a la comprobación del programa, se realizó igual manera la medición del tiempo de demora en esta labor. Para este paso se realizó la medición del proceso para la estandarización de 50 documentos distintos de la Sección Primera del Consejo de Estado, obteniendo como resultado un tiempo total de 84.75 segundos que equivale a aproximadamente 1.695 segundos por documento.

CONSEJO DE ESTADO

SALA DE LO CONTENCIOSO ADMINISTRATIVO

SECCIÓN PRIMERA

SUBSECCIÓN A

Consejero ponente: HERNANDO SÁNCHEZ SÁNCHEZ

Bogotá, D.C., diecinueve (19) de septiembre de dos mil dieciocho (2018)

Radicación número:15001-23-31-000-2003-01553-01

Actor: CENTRO DE REHABILITACIÓN INTEGRAL DE BOYACÁ E.S.E

Demandado: DEPARTAMENTO DE BOYACÁ Y ASAMBLEA DE BOYACÁ

Referencia:

Este Despacho procede a reconocer personería a los abogados Rafael Ricardo Hernández Barrera en calidad de apoderado del Departamento de Boyacá y Angélica María Medina Garavito en calidad de apoderada de la Empresa Social del Estado Centro de Rehabilitación Integral de Boyacá.

Consideraciones

1. Vistos los artículos 74 y 75 de la Ley 1564 de 12 de julio de 2012, sobre los poderes y la designación y sustitución de apoderados.
2. Atendiendo a que la abogada Angélica María Medina Garavito, identificada con la cédula de ciudadanía núm. 1.049.631.431 y con tarjeta profesional de abogado núm. 259.974, expedida por el Consejo Superior de la Judicatura, allegó poder para actuar en calidad de apoderada especial de la Empresa Social del Estado Centro de Rehabilitación Integral de Boyacá, y considerando que el poder cumple con los requisitos de ley, este Despacho le reconocerá la respectiva personería.

3. Atendiendo a que el abogado Rafael Ricardo Hernández Barrera, identificado con la cédula de ciudadanía núm. 74.382.137 y con tarjeta profesional de abogado núm. 180.354, expedida por el Consejo Superior de la Judicatura, allegó poder para actuar en calidad de apoderado especial del Departamento de Boyacá y considerando que el poder cumple con los requisitos de ley, este Despacho le reconocerá la respectiva personería.

En mérito de lo expuesto, el Despacho,

RESUELVE:

PRIMERO: Reconocer personería a la abogada Angélica María Medina Garavito identificada con la cédula de ciudadanía núm. 1.049.631.431 y con tarjeta profesional de abogado núm. 259.974, expedida por el Consejo Superior de la

FIGURA 27: Jurisprudencia estandarizada sin titular. Fuente: Autores.

Tesis:

A juicio de y para los efectos de poder adoptar una determinación sobre este punto en particular, deberá tenerse presente que desde el día 1° de enero de 1994, es decir, desde mucho antes de proferirse las resoluciones acusadas, empezó a regir 344 de del Acuerdo de Cartagena, cuyo artículo 113 introdujo modificaciones radicales a la acción especial de nulidad de registros marcarios, entre las cuales se destaca, en primer término, la exigencia de acreditar un interés directo para poder incoarla y, en segundo lugar, la supresión de la caducidad de cinco (5) años prevista en el artículo 596 del Código de Comercio. [...] En efecto, el artículo 113 de la precitada decisión comunitaria dispuso ad pedem literae o siguiente. Las acciones de nulidad que se deriven del presente artículo, podrán solicitarse en cualquier momento. (La negrilla y el subrayado son ajenos al texto). Las acciones de nulidad que se deriven del presente artículo, podrán solicitarse en cualquier momento.

Problema:

La acción de nulidad impetrada por la señora ELVIA NARIÑO DE PÉREZ, pretende la declaratoria de nulidad de las Resoluciones núms. 11218 y 11231, ambas del 30 de abril de 1997 de 1994, expedidas por de de Signos Distintivos de de Industria y Comercio, mediante las cuales se concedió a la firma ALIMENTICIOS LTDA, el registro de la marca mixta , para distinguir con ella productos y servicios comprendidos en las clases 29 y 42 de de Niza, y que, como consecuencia de tales decisiones, se ordene la cancelación de los certificados de registro núms. 196671 y 196684 expedidos por la mencionada Superintendencia. [...] Se trata de establecer, en primer término, si la marca mixta , cuyo registro fue concedido a la precitada sociedad comercial en virtud de lo dispuesto en los actos demandados, se encuentra incursa o no en alguna de las causales de irregistrabilidad establecidas en los artículos 81, 83 literal b) y 113 literales a) y c) de 344 de del Acuerdo de Cartagena, vigente al momento de proferirse las resoluciones acusadas y, en segundo término, si resulta o no procedente brindar protección jurídica al nombre comercial en los términos del artículo 128 de la referida Decisión comunitaria, cuya creación y uso fue anterior al registro de la marca que ostenta la misma denominación. [...] El signo mixto , alrededor del cual gira la presente controversia, se encuentra representado gráficamente de la siguiente manera. [...] Atendiendo las recomendaciones contenidas en la interpretación prejudicial emitida por el Tribunal de Justicia de , abordará el análisis de la temática planteada, partiendo del presupuesto de que el aspecto denominativo es en este caso el que predomina frente al gráfico, por ser el que genera una mayor recordación en el público consumidor. [...] Con todo, antes de emprender el análisis de la situación expuesta, se impone hacer un pronunciamiento de fondo con respecto a las excepciones de caducidad de la acción e ineptitud sustantiva de la demanda, propuestas por la firma ALIMENTICIOS LTDA, en su calidad de tercera interesada en las resultas del proceso. - La autoridad nacional competente podrá decretar, de oficio o a petición de parte interesada, la nulidad del registro de una marca, previa audiencia de las partes interesadas, cuando. [...] El registro se haya concedido en contravención de cualquiera de las disposiciones de la presente Decisión;.[...] El registro se hubiere otorgado con base en datos o documentos previamente declarados como falsos o inexactos por la autoridad nacional competente, contenidos en la solicitud y que sean esenciales;.[...] El registro se haya obtenido de mala fe. [...] Se

FIGURA 28: Conjunto de Párrafos de Tesis y problemas jurídicos de la jurisprudencia estandarizada. Fuente: Autores.

8.3. Encuesta de Estandarización y la titulación de las sentencias

Antes de dar paso al desarrollo de los *software*s se realizó la encuesta planteada para la estimación de los tiempos y de las capacidades de los relatores ante las labores de estandarización y titulación .En la Figura 29, Figura 30, Figura 31 y Figura 32, se puede observar las respuestas dadas por 39 relatores de la Sección Primera, para las 4 preguntas planteadas.

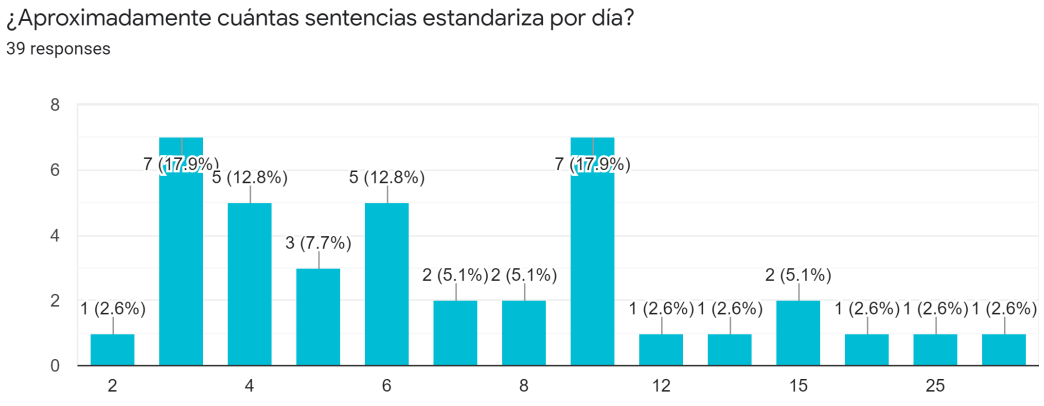


FIGURA 29: Número aproximado de sentencias estandarizadas por día. Fuente: Autores.

En aproximado, la relatoría del Consejo de Estado estandariza 8 sentencias con una tendencia a que sean 9 sentencias al día.

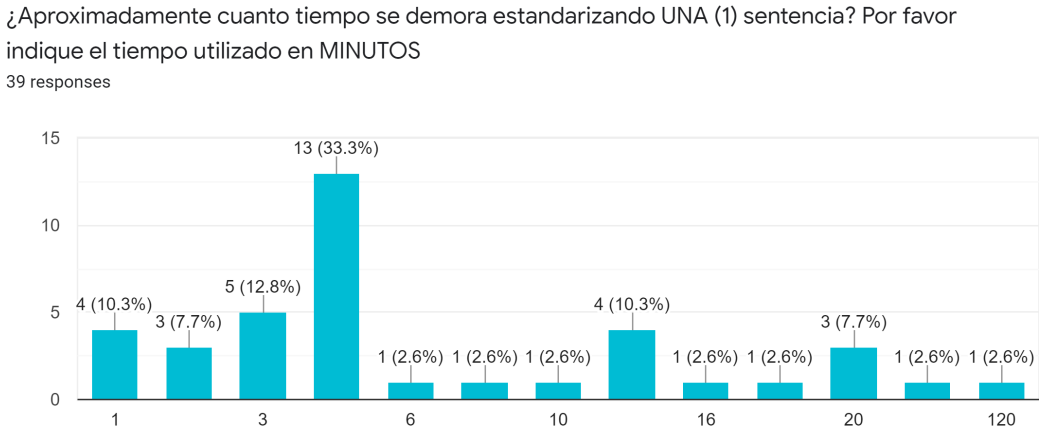


FIGURA 30: Tiempo aproximado que demora en estandarizarse 1 sentencia. Fuente: Autores.

En promedio estandariza 1 sentencia en 15 minutos con una tendencia a extenderse hasta 18 minutos por sentencia.

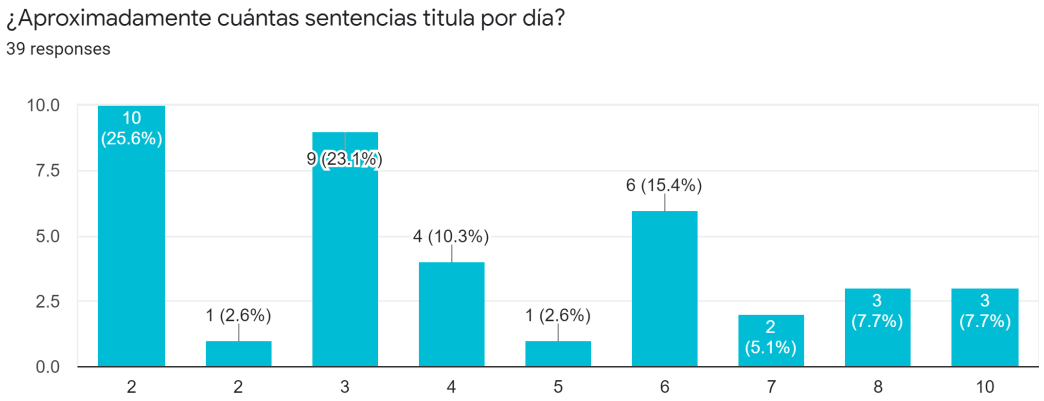


FIGURA 31: Número aproximado de sentencias tituladas al día. Fuente: Autores.

En aproximado, la relatoría del Consejo de Estado titula 4 sentencias con una tendencia a que sean 8 al día.

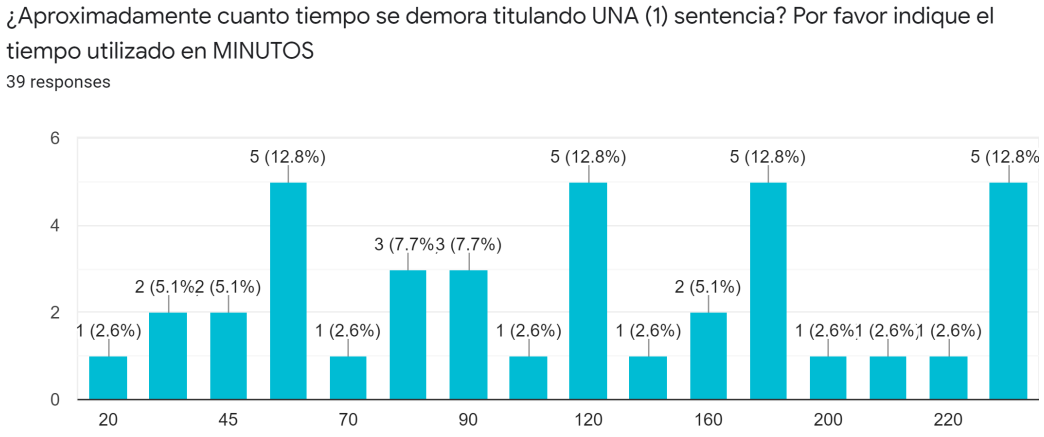


FIGURA 32: Tiempo aproximado en minutos de la titulación de una sentencia. Fuente: Autores.

En promedio se titula 1 sentencia en 127 minutos con una tendencia a extenderse hasta 160 minutos por sentencia.

Con base a las respuestas de la encuesta se puede saber que aproximadamente para el proceso de estandarización se llega a dedicar desde 120 min (2 Horas) hasta 162 min (2 Horas y 42 min)

	Aprox. de los relatores	Aprox. de los programas
Tiempo de Estandarización (segundos)	900 a 1080	0.360
Tiempo de Titulación (segundos)	7620	1.695
Sentencias por día	8 a 9	8000
Titulaciones por día	4 a 8	1700

TABLA 13: Comparación de tiempos en la labor de estandarización y titulación de los relatores y los programas diseñados.

por día, mientras que para el proceso de titulación aproximadamente por día se dedican 508 min (8 Horas y 28 min) hasta 1280 min (21 Horas y 20 min) en esta labor.

Por medio de los tiempos aproximados que puede llegar a demorar un relator en los procesos de estandarización y titulación de las sentencias, se logra dar respuesta de la eficiencia de los *software*s desarrollados. Esta comparación se muestra por medio de la Tabla 13.

Capítulo 9

Conclusiones y trabajo futuro

Tras analizar las jurisprudencias de la Sección Primera del Consejo de Estado en conjunto con los relatores y el Manual de la Relatoría se logró establecer una estructura del formato de estos documentos, estableciendo las reglas que se describieron en las secciones 7.4 y la 7.5. Por medio de la estructura establecida, se dio paso al desarrollo de una aplicación del *software* de estandarización de las jurisprudencias, para la fijación de un formato estándar y único. La correcta verificación del funcionamiento de esta aplicación se realizó por medio de múltiples pruebas con la documentación dada por el Consejo de Estado y con la aprobación de estos mimos. En ello se obtuvo como resultado una correcta configuración de los documentos para cada uno de los casos, según la reglamentación para dar paso a un único formato.

La Implementación de una base de datos basada en jurisprudencias que han sido previamente tituladas por los relatores del Consejo de Estado, se logró por medio de la ayuda dada por los relatores (Sección 7.4). Dadas estas estructuras planteadas se definió y se propuso el desarrollo de un *software* para la recolección de los datos requeridos por el algoritmo con documentos ya titulados. Este programa no solo abrió el paso al entrenamiento y la evaluación de un algoritmo enfocado en la identificación y extracción de los problemas jurídicos y las tesis en las jurisprudencias. Dadas las características del *software* su aplicación puede darse a distintas secciones del Consejo de Estado, haciendo uso de los documentos ya titulados.

El desarrollo del algoritmo enfocado en la identificación y extracción de los problemas jurídicos y las tesis en las jurisprudencias, se logró mediante técnicas de PLN definidas en la sección 6.4. El diseño del desarrollo del algoritmo se dio por medio de los 5 diferentes modelos del NLP propuestos. Para las pruebas generadas para el desarrollo de los modelos, se logró demostrar por medio de los resultados mostrados la Tabla 5 y en Tabla 6 que el

algoritmo mejor capacitado para la identificación de los problemas jurídicos y las tesis en las providencias de la Sección Primera del Consejo de Estado corresponde al modelo de N-Gramas. Este modelo logra cumplir con la exactitud esperada en comparación con el modelo de referencia (71.29 % y 70.92 %), además presenta el valor más cercano a 1 entre los algoritmos probados para las etapas de entrenamiento y de prueba (95.35 % y 94.4 %). Estos resultados confirman que es posible dar respuesta a esta labor de manera automática haciendo uso de técnicas de PLN.

Finalmente, con los resultados de la encuesta (Sección 8.3) de los tiempos aproximados que puede llegar a demorar un relator en los procesos de estandarización y titulación de las sentencias, se logra dar respuesta de la eficiencia de los *software*s desarrollados. Haciendo uso del *software* diseñado para la estandarización de los documentos de la Sección Primera del Consejo de Estado se logra esta labor en aproximadamente 360 ms por documento, que equivale a ser 2500 veces más rápido, que el tiempo que puede demorar un relator. Del mismo modo, se usó el segundo *software* diseñado para la titulación de los documentos, obteniendo una respuesta de esta labor en aproximadamente 1.695 s por documento, que equivale a ser 74.926 veces más rápido que un relator en este proceso. Para ambos casos significa una respuesta eficiente para apoyar en el proceso de titulación.

Dada la metodología que se planteó, las consultas solicitadas y las especificaciones de funcionalidad requeridas en los *software*s, se logró dar respuesta al problema de estandarización de las jurisprudencias, extracción y clasificación de forma automática de los problemas y las tesis jurídicas en las providencias de la Sección Primera del Consejo de Estado.

Este proyecto entrega como resultado dos diferentes herramientas de software, una para la estandarización del formato (Figura 13) y otra para la titulación automática de las providencias en la relatoría del Consejo de Estado basada en el uso de técnicas de Procesamiento del Lenguaje Natural (NLP) e Inteligencia Artificial (Figura 13).

La finalidad de este proyecto es agilizar y descongestionar los procesos relacionados a la asistencia judicial, llevados a cabo por las diferentes secciones del Consejo de Estado, los resultados obtenidos en la Tabla 5 y la Tabla 6 demuestran que su implementación en el Consejo de Estado podrá mejorar el acceso a este servicio reduciendo el tiempo de espera. Al reducir drásticamente los tiempos de respuesta, el ciudadano puede tener una claridad básica y pronta sobre la temática de su solicitud. Los resultados y el procedimiento desarrollado permiten también afirmar que, por medio del algoritmo, es posible dar mejora a un entorno

más general de la justicia ante la versatilidad que presenta el sistema, permitiendo su desarrollo a diferentes áreas de las planteadas.

A la vez, permite descongestionar las diferentes secciones del Consejo de Estado de trabajo acumulado y no resuelto a lo largo de los años. El desarrollo y la evaluación de distintos algoritmos basados en técnicas de NLP para clasificar Jurisprudencias y titularlas de manera precisa, permite ser ajustado a diferentes épocas de tiempo y a la varianza que se encuentran en las jurisprudencias conforme la legislación jurídica colombiana avanza.

Sin embargo, como trabajo futuro, se plantea la posibilidad de integrar un sistema de aprendizaje por refuerzo continuo para analizar la variación de la estructura estándar de las jurisprudencias y así obtener un sistema robusto. Este a su vez sirve como base del modelo de titulación, al poder aplicar un mayor número de características que permitan la mínima interacción humana y la mayor precisión ante la relatoría del Consejo de Estado.

Bibliografía

[1] Amory Reyes. *Machine Learning: Qué es, para qué sirve - Atica Ingeniería*. Feb. de 2020. URL: https://aticaingenieria.cl/machine-learning-que-es-para-que-sirve/.

[2] Gustavo A Cardona y Juan M Calderon. «Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations». En: *Applied Sciences* 9.8 (2019), pág. 1702.

[3] Jose León y col. «Robot swarms theory applicable to seek and rescue operation». En: *International Conference on Intelligent Systems Design and Applications*. Springer. 2016, págs. 1061-1070.

[4] Wilson O Quesada y col. «Leader-follower formation for uav robot swarm based on fuzzy logic theory». En: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2018, págs. 740-751.

[5] Nicolás Gómez y col. «Leader-follower Behavior in Multi-agent Systems for Search and Rescue Based on PSO Approach». En: *SoutheastCon 2022*. IEEE. 2022, págs. 413-420.

[6] Juan D Pabon y col. «Event-Triggered Control for Weight-Unbalanced Directed Robot Networks». En: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, págs. 5831-5836.

[7] Nestor I Ospina y col. «Argrohbots: An affordable and replicable ground homogeneous robot swarm testbed». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 256-261.

[8] Edgar C Camacho, Nestor I Ospina y Juan M Calderón. «COVID-Bot: UV-C Based Autonomous Sanitizing Robotic Platform for COVID-19». En: *Ifac-papersonline* 54.13 (2021), págs. 317-322.

[9] Edgar C Camacho, Jose Guillermo Guarnizo, Juan M Calderon y col. «Design and Construction of a Cost-Oriented Mobile Robot for Domestic Assistance». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 293-298.

[10] GA Cardona y col. «Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning». En: *2019 SoutheastCon*. IEEE. 2019, págs. 1-7.

[11] Laura J Padilla Reyes y col. «Adaptable Recommendation System for Outfit Selection with Deep Learning Approach». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 605-610.

[12] Saith Rodrıguez y col. «Fast path planning algorithm for the robocup small size league». En: *Robot Soccer World Cup*. Springer. 2014, págs. 407-418.

[13] Saith Rodrıguez y col. «STOx's 2016 Team description paper». En: (2013).

[14] Saith Rodrıguez y col. «STOx's 2015 Extended Team Description Paper». En: *Joao Pessoa, Brazil* (2014).

[15] Juan M Calderon y col. «A Robot soccer team as a strategy to develop educational iniciatives». En: *Latin American and Caribbean Conference for Engineering and Technology, Panama City, Panama*. 2012.

[16] Heyson Báez y col. «Application of an educational strategy based on a soccer robotic platform». En: *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE. 2013, págs. 1-6.

[17] Marta Lucía y col. «CONSEJO NACIONAL DE POLÍTICA ECONÓMICA Y SOCIAL CONPES Iván Duque Márquez Presidente de la República». En: ().

[18] Sagee Geetha Sethu. «The Inevitability of an International Regulatory Framework for Artificial Intelligence». En: Institute of Electrical y Electronics Engineers Inc., abr. de 2019, págs. 367-372. ISBN: 9781538680100. DOI: 10.1109/ICACTM.2019.8776819.

[19] Harry Surden. *Machine Learning and Law*. Mar. de 2014. URL: https://papers.ssrn.com/abstract=2417415.

[20] CONSEJO DE ESTADO. *ACUERDO 80 DE 2019*. 2019. URL: https://normativa.colpensiones.gov.co/colpens/docs/pdf/acuerdo_cdeestado_0080_2019.pdf.

[21] Archit P. Patil, Devansh Jain Nawal y Dipika Jain. «Crime Prediction Application Using Artificial Intelligence». En: vol. 605. Springer, 2020, págs. 238-245. ISBN: 9783030305765. DOI: 10.1007/978-3-030-30577-2_20. URL: https://link.springer.com/chapter/10.1007/978-3-030-30577-2_20.

[22] Brian Seamus Haney. «Applied Natural Language Processing for Law Practice». En: *SSRN Electronic Journal* (nov. de 2019). ISSN: 1556-5068. DOI: 10.2139/ssrn.3476351. URL: https://papers.ssrn.com/abstract=3476351.

[23] M Palmirani. *Legal Knowledge and Information Systems: JURIX 2018: The Thirty-first Annual Conference*. IOS Press, 2018. ISBN: 9781614999355. URL: https://books.google.com.co/books?id=hrh9DwAAQBAJ.

[24] Shuijing Hu. «Comparative study on patent eligibility of artificial intelligence in the united states, china and japan». En: Institute of Electrical y Electronics Engineers Inc., ene. de 2020, págs. 965-968. ISBN: 9781728166971. DOI: 10.1109/ICITBS49701.2020.00215.

[25] Jiajing Li y col. «A Markov logic networks based method to predict judicial decisions of divorce cases». En: Institute of Electrical y Electronics Engineers Inc., oct. de 2018, págs. 129-132. ISBN: 9781538651827. DOI: 10.1109/SmartCloud.2018.00029.

[26] Ge Yan y col. «Law Article Prediction Based on Deep Learning». En: Institute of Electrical y Electronics Engineers Inc., jul. de 2019, págs. 281-284. ISBN: 9781728139258. DOI: 10.1109/QRS-C.2019.00060.

[27] Masha Medvedeva, Michel Vols y Martijn Wieling. «Using machine learning to predict decisions of the European Court of Human Rights». En: *Artificial Intelligence and Law* 28 (2 2020), págs. 237-266. ISSN: 15728382. DOI: 10.1007/s10506-019-09255-y. URL: https://doi.org/10.1007/s10506-019-09255-y.

[28] Baogui Chen y col. «A Deep Learning Method for Judicial Decision Support». En: Institute of Electrical y Electronics Engineers Inc., jul. de 2019, págs. 145-149. ISBN: 9781728139258. DOI: 10.1109/QRS-C.2019.00040.

[29] Chunyu Xia y col. «Similarity Analysis of Law Documents Based on Word2vec». En: Institute of Electrical y Electronics Engineers Inc., jul. de 2019, págs. 354-357. ISBN: 9781728139258. DOI: 10.1109/QRS-C.2019.00072.

[30] Yadong Cui, Cao Yan y Liu Yan. *Artificial intelligence and judicial modernization*. Springer Singapore, ene. de 2019, págs. 1-224. ISBN: 9789813298804. DOI: 10.1007/978-981-32-9880-4.

[31] Reshma Sheik y S. Jaya Nirmala. «Deep Learning Techniques for Legal Text Summarization». En: *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering, UPCON 2021* (2021). DOI: 10.1109/UPCON52273.2021.9667640.

[32] Naimoonisa Begum y Ankur Goyal. «Analysis of Legal Case Document Automated Summarizer». En: *Proceedings of IEEE International Conference on Signal Processing,Computing and Control* 2021-October (2021), págs. 533-538. ISSN: 26438615. DOI: 10.1109/ISPCC53510.2021.9609442.

[33] Purbid Bambroo y Aditi Awasthi. «LegalDB: Long DistilBERT for Legal Document Classification; LegalDB: Long DistilBERT for Legal Document Classification». En: (2021). DOI: 10.1109/ICAECT49130.2021.9392558. URL: https://www.courtlistener.com/api/bulk-.

[34] Dinh-Truong Do. «Kodiak@ALQAC2021: Deep Learning for Vietnamese Legal Information Processing». En: (dic. de 2021), págs. 1-5. DOI: 10.1109/KSE53942.2021.9648744.

[35] Daniel A. Rincón-Riveros y col. «Automation system based on NLP for legal clinic assistance». En: *IFAC-PapersOnLine* 54 (13 2021), págs. 283-288. ISSN: 24058963. DOI: 10.1016/J.IFACOL.2021.10.460.

[36] Riya Sil y col. «Artificial Intelligence and Machine Learning based Legal Application: The State-of-the-Art and Future Research Trends». En: vol. 2019-Janua. Institute of Electrical y Electronics Engineers Inc., oct. de 2019, págs. 57-62. ISBN: 9781728148267. DOI: 10.1109/ICCCIS48478.2019.8974479.

[37] Ali Sadeghian y col. «Automatic semantic edge labeling over legal citation graphs». En: *Artificial Intelligence and Law* 26 (2018), págs. 127-144. DOI: 10.1007/s10506-018-9217-1. URL: https://doi.org/10.1007/s10506-018-9217-1.

[38] Kirill Romanyuk. «Game Theoretic Approach for Applying Artificial Intelligence in the Credit Industry». En: Institute of Electrical y Electronics Engineers Inc., feb. de 2019, págs. 1-6. ISBN: 9781538671467. DOI: 10.1109/CTIT.2018.8649493.

[39] R.S. Becerra. *La responsabilidad extracontractual de la administración pública*. Ediciones Jurıdicas Gustavo Ibáñez, 2003. ISBN: 9789588192185.

[40] Lambrini Seremeti y Ioannis Kougias. «Legal issues within ambient intelligence environments». En: Institute of Electrical y Electronics Engineers Inc., jul. de 2019. ISBN: 9781728149592. DOI: 10.1109/IISA.2019.8900748.

[41] Harry Surden. «Artificial Intelligence and Law: An Overview, 35 GA». En: *U. L. Rev* (2019). URL: https://scholar.law.colorado.edu/articles/1234/.

[42] Kevin D. Ashley. *Artificial intelligence and legal analytics: New tools for law practice in the digital age*. Cambridge University Press, ene. de 2017, págs. 1-426. ISBN: 9781316761380. DOI: 10.1017/9781316761380. URL: /core/books/artificial-intelligence-and-legal-analytics/E7D705EEF392501A1DB180645917E7E0.

[43] Stephan Raaijmakers. «Artificial Intelligence for Law Enforcement: Challenges and Opportunities». En: *IEEE Security and Privacy* 17 (5 sep. de 2019), págs. 74-77. ISSN: 15584046. DOI: 10.1109/MSEC.2019.2925649.

[44] Juan, Jorge Almonacid Sierra y Yeisson Coronel Ávila. «Aplicabilidad de la inteligencia artificial y la tecnología blockchain en el derecho contractual privado * Application of Artificial Intelligence and Blockchain in Contract Law». En: *Revista de derecho privado* (2020), pág. 24. DOI: 10.18601/01234366.

[45] Alfredo Monroy, Hiramand Calvo y Alexander Gelbukh. «NLP for shallow question answering of legal documents using graphs». En: vol. 5449 LNCS. Springer, Berlin, Heidelberg, 2009, págs. 498-508. ISBN: 3642003818. DOI: 10.1007/978-3-642-00382-0_40. URL: https://link.springer.com/chapter/10.1007/978-3-642-00382-0_40.

[46] Marco Lippi y col. «CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service». En: *Artificial Intelligence and Law* 27 (2019), págs. 117-139. DOI: 10.1007/s10506-019-09243-2. URL: https://doi.org/10.1007/s10506-019-09243-2.

[47] Ilaria Angela Araszkiewicz Michałand Amantea y col. «ICAIL Doctoral Consortium, Montreal 2019». En: vol. 28. Springer, jun. de 2020, págs. 267-280. DOI: 10.1007/s10506-020-09267-z. URL: https://link-springer-com.crai-ustadigital.usantotomas.edu.co/article/10.1007/s10506-020-09267-z.

[48] John Zeleznikow y Andrew Stranieri. «Knowledge discovery in the legal domain». En: IEEE, 1997, págs. 574-577. DOI: 10.1109/tai.1997.632305.

[49] *Office Open XML - Anatomy of an OOXML WordProcessingML File*. URL: http://officeopenxml.com/anatomyofOOXML.php.

[50] Jasmeet Singh y Vishal Gupta. «Text Stemming». En: *ACM Computing Surveys (CSUR)* 49 (3 sep. de 2016). ISSN: 15577341. DOI: 10.1145/2975608. URL: https://dl.acm.org/doi/abs/10.1145/2975608.

[51] Vijayarani Mohan. «Preprocessing Techniques for Text Mining-An Overview Privacy Preserving Data Mining View project». En: (). URL: https://www.researchgate.net/publication/339529230.

[52] *What Are Baseline Models and Benchmarking For Machine Learning… – Towards AI*. URL: https://towardsai.net/p/l/what-are-baseline-models-and-benchmarking-for-machine-learning-why-we-need-them.

[53] Jurafsky Daniel y James H Martin. «Speech and Language Processing». En: (2021).

[54] Cai Zhi Liu y col. «Research of Text Classification Based on Improved TF-IDF Algorithm». En: *2018 IEEE International Conference of Intelligent Robotic and Control Engineering, IRCE 2018* (oct. de 2018), págs. 69-73. DOI: 10.1109/IRCE.2018.8492945.

[55] *1.4. Support Vector Machines — scikit-learn 1.0.2 documentation*. URL: https://scikit-learn.org/stable/modules/svm.html.

[56] *6.1. Pipelines and composite estimators — scikit-learn 1.0.2 documentation*. URL: https://scikit-learn.org/stable/modules/compose.html#combining-estimators.

[57] *1.13. Feature selection — scikit-learn 1.0.2 documentation*. URL: https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection.

[58] Leo Breiman. «Random Forests». En: 45 (2001), págs. 5-32.

[59] *Consejo de Estado | – Justicia – Guía – Control*. URL: https://www.consejodeestado.gov.co/.

[60] Larry. *Python - Transformar números a literales*. Mar. de 2008. URL: https://www.lawebdelprogramador.com/foros/Python/297286-Transformar-numeros-a-literales.html.

Anexos

Anexo 1. Código del programa utilizado para la estandarización de las jurisprudencias.

Código utilizado para transformar de números a letras, tomado de "Transformar números a literales". [60]

```
def numero_to_letras(numero):
    indicador = [("","("MIL","MIL"),("MILLON","MILLONES"),("MIL","MIL"),("BILLON","BILLONES
    ")]
    entero = int(numero)
    decimal = int(round((numero - entero)*100))
    #print 'decimal : ',decimal
    contador = 0
    numero_letras = ""
    while entero >0:
        a = entero % 1000
        if contador == 0:
            en_letras = convierte_cifra(a,1).strip()
        else :
            en_letras = convierte_cifra(a,0).strip()
        if a==0:
            numero_letras = en_letras+""+numero_letras
        elif a==1:
            if contador in (1,3):
                numero_letras = indicador[contador][0]+" "+numero_letras
            else:
                numero_letras = en_letras+" "+indicador[contador][0]+" "+numero_letras
        else:
            numero_letras = en_letras+" "+indicador[contador][1]+" "+numero_letras
        numero_letras = numero_letras.strip()
        contador = contador + 1
        entero = int(entero / 1000)
    #numero_letras = numero_letras+" con " + str(decimal) +"/100"
    #print('numero: ',numero)
```

```
    #print(numero_letras)
    return numero_letras
def convierte_cifra(numero,sw):
   lista_centana = ["",("CIEN","CIENTO"),"DOSCIENTOS","TRESCIENTOS","CUATROCIENTOS","
    QUINIENTOS","SEISCIENTOS","SETECIENTOS","OCHOCIENTOS","NOVECIENTOS"]
   lista_decena = ["",("DIEZ","ONCE","DOCE","TRECE","CATORCE","QUINCE","DIECISEIS","DIECISIETE
    ","DIECIOCHO","DIECINUEVE"),
                   ("VEINTE","VEINTI"),("TREINTA","TREINTA Y "),("CUARENTA" , "CUARENTA Y "),
                   ("CINCUENTA" , "CINCUENTA Y "),("SESENTA" , "SESENTA Y "),
                   ("SETENTA" , "SETENTA Y "),("OCHENTA" , "OCHENTA Y "),
                   ("NOVENTA" , "NOVENTA Y ")]
   lista_unidad = ["",("UN","UNO"),"DOS","TRES","CUATRO","CINCO","SEIS","SIETE","OCHO","NUEVE"
    ]
   centena = int (numero / 100)
   decena = int((numero -(centena * 100))/10)
   unidad = int(numero - (centena * 100 + decena * 10))
   #print "centena: ",centena, "decena: ",decena,'unidad: ',unidad
   texto_centena = ""
   texto_decena = ""
   texto_unidad = ""
   #Validad las centenas
   texto_centena = lista_centana[centena]
   if centena == 1:
       if (decena + unidad)!=0:
           texto_centena = texto_centena[1]
       else :
           texto_centena = texto_centena[0]

   #Valida las decenas
   texto_decena = lista_decena[decena]
   if decena == 1 :
       texto_decena = texto_decena[unidad]
   elif decena > 1:
       if unidad != 0 :
           texto_decena = texto_decena[1]
       else:
           texto_decena = texto_decena[0]
   #Validar las unidades
   #print "texto_unidad: ",texto_unidad
   if decena != 1:
       texto_unidad = lista_unidad[unidad]
       if unidad == 1:
           texto_unidad = texto_unidad[sw]
   return "%s %s %s" %(texto_centena,texto_decena,texto_unidad)
```

Código del programa utilizado para la estandarización de las jurisprudencias, generado con ayuda de la herramienta de Qt Designer para el diseño de la interfaz.

```
# -*- coding: utf-8 -*-
```

```
# Form implementation generated from reading ui file 'Formato.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.
from docx import Document
import os
from os import listdir,getcwd
from os.path import isfile, join
from docx.shared import Pt
from docx.shared import Cm
import shutil
from win32com import client as wc
import re
import docx
from difflib import SequenceMatcher as SM
from docx.enum.text import WD_ALIGN_PARAGRAPH
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5 import uic
from PyQt5.QtWidgets import QAction,QFileDialog
from PyQt5.QtGui import QIcon
import sys
import numpy as np

def IdMes(Texto):
    for i,M in enumerate(Meses):
        if M in Texto.lower():
            return MesesC[i]
    else: return ""

def AllNumber(string):
    Por=[1 for C in string if C.isnumeric()]
    if np.sum(Por)/len(string)>0.7:
        return True
    else:return False

Meses=["ene","feb","mar","abr","may","jun","jul","ago","sep","nov","dic","set","agt"]
MesesC=["enero","febrero","marzo","abril","mayo","junio","julio","agosto","septiembre","
    noviembre","diciembre","septiembre","agosto"]

def IsFolderPath(Path_W):
    global Formated_docx_Files
    global docx_Files
    global doc_Files

    P1=Path_W +  "/Formated_docx_Files/"
    P2=Path_W +  "/docx_Files/"
    P3=Path_W +  "/doc_Files/"

    if not os.path.exists(P1):
```

```
        os.makedirs(P1)
    if not os.path.exists( P2):
        os.makedirs(P2)
    if not os.path.exists(P3 ):
        os.makedirs(P3)

    Formated_docx_Files = P1
    docx_Files = P2
    doc_Files =P3



def IsFolder():


    if not os.path.exists('OG_Files'):
        os.makedirs('OG_Files')
        print('OG_Files directory created!  \n')
    if not os.path.exists('doc_Files'):
        os.makedirs('doc_Files')
        print('doc_Files directory created!  \n')
    if not os.path.exists('docx_Files'):
        os.makedirs('docx_Files')
        print('docx_Files directory created!  \n')
    if not os.path.exists('Formated_docx_Files'):
        os.makedirs('Formated_docx_Files')
        print('Formated_docx_Files directory created!  \n')

    if not os.path.exists('doc97_Out'):
        os.makedirs('doc97_Out')
        print('doc97_Out directory created!  \n')
#Formated_docx_Files = "C:/Users/willi/Desktop/Formato/A"
#docx_Files = "C:/Users/willi/Desktop/Formato/"
#doc_Files = "C:/Users/willi/Desktop/Formato/C"
global Path
global Path2
Path="     "
Path2="     "
def delete_paragraph(paragraph):
    p = paragraph._element
    p.getparent().remove(p)
    p._p = p._element = None
def DocumetArrangement():
    global Path
    for filename in os.listdir(Path):
        if filename.endswith('.doc'):
            print ('archivos .doc en la carpeta: ' + filename + ' \n')
            shutil.move(str(Path) + str(filename), str(doc_Files) + str(filename))

    for filename in os.listdir(Path):
        if filename.endswith('.docx'):
```

```
            print ('archivos .docx en la carpeta: ' + filename + ' \n')
            shutil.move(str(Path) + str(filename), str(docx_Files) + str(filename))
def doc2docx(doc_path,docx_path):
    # word = wc.DispatchEx("Word.Application")
    word = wc.Dispatch('word.Application')
    doc = word.Documents.Open(doc_path)
    doc.SaveAs(docx_path,16)  #16 doc2docx
    doc.Close()
    word.Quit()
def doc_Docx_Converter():

    for filename in os.listdir(doc_Files):
        if filename.endswith('.doc'):
            doc_path = doc_Files + filename
            docx_path = docx_Files + filename + '_converted.docx'
            print ('archivo .doc en la carpeta: ' + filename + ' \n')
            print('pathway ' + doc_path)

            print('preparando conversion a *.docx' ' \n')
            #word = wc.DispatchEx("Word.Application")
            doc2docx(doc_path,docx_path)
            print('Archivo '+ docx_path + ' convertido exitosamente a *.docx  \n')

        if filename.endswith('.docx'):
            print('preparando para mover archivo a la carpeta' + docx_Files + ' \n')
            print ('archivos .docx en la carpeta: ' + filename + ' \n')
            shutil.move(str(doc_Files) + str(filename), str(docx_Files) + str(filename))
            print ('archivos .docx movidos a la carpeta: ' + docx_Files + ' \n')
#Lista de archivos .docx en la carpeta actual
def ToListdocx():
    global Formated_docx_Files
    global docx_Files
    global doc_Files
    ruta = docx_Files
    return [arch for arch in listdir(ruta) if (isfile(join(ruta, arch))) and ".docx" in arch]
def replaceheader(document):
    # HEADERS:
    section = document.sections[0]
    header = section.header
    header.is_linked_to_previous = True

def AllBold(paragraph):
    for run in paragraph.runs:
        run.font.bold=True

#Lectura del documento y conversion a lista de parrafos
def docxFormatter(self,docs_file_name):
    global Path
    global Formated_docx_Files
    global docx_Files
    global doc_Files
```

```
progress=0
print('number of .docx files in path: ' + str(len(docs_file_name))+ ' \n')
print('.docx files in path: ' + str(docs_file_name) + ' \n')
#print("\n")
docNum = 0
for ND,file in enumerate(docs_file_name):
    document = Document(docx_Files + file)
    print("Documento:",file,"========================================")
    for i,P in enumerate(document.paragraphs):
        paragraph_format = P.paragraph_format
        tab_stops = paragraph_format.tab_stops
    for i,P in enumerate(document.paragraphs):
        P.text = P.text.replace(' \t',"")
    print('El numero de tabs es: ', len(tab_stops))
    if len(tab_stops) > 0:
        tab_stops.clear_all()
        print('tabs removidos')
    ParCounter = 0
    Datos={}
    Referencia=""
    count=0
    FindDate=False
    FindRad=False
    for i,P in enumerate(document.paragraphs):

        #try:

            if len(P.text)>3 and count<4:
                if not FindDate:
                    IC=P.text.rfind(",")
                    CNum=bool(re.search(r'\d{4}',P.text))

                    if 50>IC>1:
                        Ciudad=P.text[:IC]
                        D = re.search(r'\d\d?',P.text)
                        Mes=IdMes(P.text)
                        A=re.search(r'\d{4}',P.text)
                        if D is not None and A is not None:
                            Dia = D.group()
                            DiaLetras=numero_to_letras(int(Dia))
                            Año=A.group()
                            AñoLetras=numero_to_letras(int(Año))
                            Datos['Fecha']=Ciudad+","+ DiaLetras.lower()+ "("+ str(Dia)
+")"+" de " + Mes.lower()+ " de "+ AñoLetras.lower() +"("+ str(Año) +")"
                            FindDate=True

                IP=P.text.find(":")
                if IP>0:
                    SuccessC=False
                    if count==0 and not AllNumber(P.text[IP+1:]):
                        Datos['Consejero']=P.text[IP+1:]
```

```
                        count=count+1
                        SuccessC=True

                    elif count==0 and FindDate:
                        count=count+1

                    if count==1 and not SuccessC:

                        if not FindRad: m=re.search(r'\d{5}',P.text)
                        else:m=None

                        if m!=None:
                            LN= [i for i in list(reversed(range(len(P.text)))) if P.text[i
].isnumeric()]
                            Datos['Rad']=P.text[IP+1:LN[0]+1]
                            count=count+1
                            FindRad=True
                            if len(P.text[LN[0]+1:])>5:
                                Datos['Demandado']=" "
                        else:
                            Referencia=P.text[IP+1:]
                    elif count==2:
                        Datos['Actor']=P.text[IP+1:]
                        count=count+1
                    elif count==3 and 'Demandado' in Datos:
                        count=count+1
                    elif count==3:
                        Datos['Demandado']=P.text[IP+1:]
                        count=count+1

    #    except:
    #        print("Fallo en la busqueda del encabezado")

    if 'Consejero' not in Datos:
        Datos['Consejero']=" "
    if 'Fecha' not in Datos:
        Datos['Fecha']=" "
    if 'Rad' not in Datos:
        Datos['Rad']=" "
    if 'Actor' not in Datos:
        Datos['Actor']=" "
    if 'Demandado' not in Datos:
        Datos['Demandado']=" "
    Datos['Actor'] = Datos['Actor'].replace("  ", "")
    Datos['Demandado'] = Datos['Demandado'].replace("  ", "")

    print('El dictado es :', Datos)


    Find=False
    for i,P in enumerate(document.paragraphs):
```

```
        m=re.search(r'\d{5}\s?\d{2}\s?\d{2}\s?\d{3}\s?\d{4}\s?\d{5}\s?\d{2}',P.text)
        if m!=None and not Find:
            Num=m.group()
            Num=re.sub(r'\s','',Num)
            othernum=Num
            print('Antes de las Correcciones: ',P.text)
            print('Primera verificacion: ',P.text.replace(P.text,othernum))
            P.text = P.text.replace(P.text,othernum)
            NewNum=Num[:5]+"-"+Num[5:7]+"-"+Num[7:9]+"-"+Num[9:12]+"-"+Num[12:16]+"-"+Num
[16:21]+"-"+Num[21:23]
            print("Coincidencia",Num,"===",NewNum)
            print('Last verify: ',P.text.replace(Num,NewNum))
            P.text = P.text.replace(Num,NewNum)
            Datos['Rad'] = P.text
            Find=True
    for i,P in enumerate(document.paragraphs):
        if len(P.text) >= 150:
            break
        delete_paragraph(P)
    for i,P in enumerate(document.paragraphs):
        AllBold(P.insert_paragraph_before(text='CONSEJO DE ESTADO', style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='SALA DE LO CONTENCIOSO ADMINISTRATIVO',
style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='SECCIÓN PRIMERA', style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='SUBSECCIÓN A', style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='Consejero ponente:' + str.upper(Datos['
Consejero']), style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        P.insert_paragraph_before(text=Datos['Fecha'], style=None)
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='Radicación número:' + Datos['Rad'], style=
None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='Actor:' + str.upper(Datos['Actor']), style
=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='Demandado:' + str.upper(Datos['Demandado'
]), style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text='Referencia:'+ Referencia, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        AllBold(P.insert_paragraph_before(text=None, style=None))
        if i == 0:
```

```
            break
    for P in document.paragraphs:

        P.paragraph_format.line_spacing = 1
        for run in P.runs: #each run is a paragraph "run"
            font = run.font
            run.font.name = 'Arial'
            run.font.size = Pt(12)
            run.font

        #changing the page margins
        sections = document.sections
        for section in sections:
            cms = 3
            #font.size = Pt(12)
            section.top_margin = Cm(cms)
            section.bottom_margin = Cm(cms)
            section.left_margin = Cm(cms)
            section.right_margin = Cm(cms)
            section.page_height = Cm(33.02)
            section.page_width = Cm(21.59)

    for i,P in enumerate(document.paragraphs):
       P.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.CENTER
       if i == 9:
            P.paragraph_format.alignment = WD_ALIGN_PARAGRAPH.JUSTIFY
            break
    RunCounter = 0
    for P in document.paragraphs:
        for run in P.runs:
            font = run.font
            #run.font.bold = True
            run.font
            RunCounter = RunCounter + 1
            if RunCounter == 1:
                break

    for P in document.paragraphs:
        P.paragraph_format.space_after = Pt(0)
        P.paragraph_format.space_before = Pt(0)
    replaceheader(document)
    docNum = docNum+1
    print('Processed Document #' + str(docNum) + ' \n')
    document.save(Formated_docx_Files + file) #saves or rewrites the document as doc &
docx

    progress=int(((ND+1)*100)/len(docs_file_name))
    #print(ND,progress,len(docs_file_name))
    self.progressBar.setProperty("value", progress)
```

```
#interface=================================================
#Codigo generado con la herramienta de Qt Designer==========
#============================================================
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("Convertidor de Formato")
        MainWindow.resize(598, 341)
        MainWindow.setWindowIcon(QIcon("CE.ico"))
        MainWindow.setWindowTitle("Auto Formato de Sentencias")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(170, 180, 241, 51))
        self.pushButton.setObjectName("pushButton")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 90, 211, 31))
        self.label.setObjectName("label")
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setGeometry(QtCore.QRect(30, 250, 561, 31))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")
        self.toolButton = QtWidgets.QToolButton(self.centralwidget)
        self.toolButton.setGeometry(QtCore.QRect(420, 100, 45, 31))
        self.toolButton.setObjectName("toolButton")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(110, 100, 301, 31))
        self.lineEdit.setObjectName("lineEdit")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(10, 10, 581, 21))
        font = QtGui.QFont()
        font.setFamily("Arial Black")
        font.setBold(True)
        font.setWeight(75)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(10, 60, 581, 21))
        self.label_4.setObjectName("label_4")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 598, 22))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(MainWindow)
```

```
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        #Botones
        self.pushButton.clicked.connect(self.on_click)
        self.pushButton.setEnabled(False)
        self.toolButton.clicked.connect(self.SearchDOC)

        #Texto Path
        self.lineEdit.setEnabled(False)



    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Conversor de Formato"))
        self.pushButton.setText(_translate("MainWindow", "Dar Formato"))
        self.label.setText(_translate("MainWindow", "Origen"))
        self.toolButton.setText(_translate("MainWindow", "..."))
        self.label_3.setText(_translate("MainWindow", "Auto Generador de Formato - Consejo de
    Estado"))
        self.label_4.setText(_translate("MainWindow", "Formato de Documento"))

    def on_click(self):
        #program RunTime
        ##################################
        #IsFolder()
        DocumetArrangement()
        doc_Docx_Converter()
        docxFormatter(self,ToListdocx())
        ##################################

    def Print(self ):
        print("HOLA")

    def SearchDOC(self):
        global Path
        global Path2

        Path =str( QFileDialog.getExistingDirectory(None, "Open a folder", 'C:/', QFileDialog.
    ShowDirsOnly))+"/"
        self.lineEdit.setText(Path)
        print(Path)
        if os.path.exists(Path) :
            IsFolderPath(Path)
            self.pushButton.setEnabled(True)




if __name__ == "__main__":
```

```
import sys
app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.show()
sys.exit(app.exec_())
```

Anexo 2. Código del programa utilizado para la extracción de la información de las titulaciones y generación de la base de datos.

```
# -*- coding: utf-8 -*-
"""
@author: william
"""

#============================================================================
#============================       Librerias        ========================
#============================================================================

import docx
import re

import numpy as np
import pandas as pd

from os import listdir
from os.path import isfile, join
from collections import OrderedDict
from difflib import SequenceMatcher as SM


#============================================================================
#====================       Direcciones carpetas        ====================
#============================================================================

#FolderP="C:\/Users\/willi\/Desktop\/NLP_Pruebas\/"
FolderP="C:\/Users\/willi\/OneDrive\/Escritorio\/NLP_Pruebas\/"
FolderS="DOCX21_F"
Path= FolderP+FolderS+"\\"
Filename="Dataset.csv"


#============================================================================
#============================        Leer Docx        ========================
#============================================================================

def ToList(ruta = Path):
    return [arch for arch in listdir(ruta) if (isfile(join(ruta, arch))) and ".docx" in arch
    and "~" not in arch]


#============================================================================
#======================       Buscar Titulación        ======================
#============================================================================

def IdTitulacion(Paragraphs):
```

```
    iend=1
    String="consejo de estado"
    for P in Paragraphs:
            for run in P.runs:
                if run.bold and String in P.text.lower() and len(P.text)<20:
                    iend=Paragraphs.index(P)
                    return iend
    return -1


#============================================================================
#=========================       Extraer Tesis        =======================
#============================================================================

def Finder(Doc,Tit,Prob):

    Indexs=[]
    IndexsP=[]
    S=[]

    #Separar Ideas en la Titulación ==========================================

    for P in Tit:
        if len(P)>5:
            T=re.sub(r'(?:\[|\()([^a-zA-Z0-9]+)(?:\]|\))','.', P)
            T=re.sub(r'[^\x00-\xFF]+',' ', T)
            T=re.sub(r'\.+\s+\.+\s?\.*\s?','.', T)
            S.append(T)

    TesisN=list(S[1].split('.'))


    #Descartar el problema de las ideas ======================================

    for k,Ti in enumerate(TesisN):
        if len(Ti)>5:
            for Pr in Prob:
                if len(Pr)>5:
                    if SM(None, Pr.lower(), Ti.lower()).ratio()>0.75:
                        IndexsP.append(k)

    #Vacio y organizar =======================================================

    if len(IndexsP)<1:
        IndexsP.append(-1)
    IndexsP=sorted(set(IndexsP))

    # Buscar Parrafos con la Tesis ===========================================

    for j,L in enumerate(TesisN):
        if len(L)>5 and j not in IndexsP:
```

```
            Ixvalue=[[i , SM(None,L.lower(),P.lower()).ratio()] for i,P in enumerate(Doc) if
    TrueIndex(L,P)]
            a=np.array(Ixvalue)

            if len(Ixvalue)>=1:
                Imax=np.argmax(a, axis=0)
                Indexs.append(int(a[Imax[1]][0]))

    #print(Indexs)
    if len(Indexs)<1:
        return [-1]

    return Indexs


#Condiciones de similitud: ==================================================

def TrueIndex(L,P):
    if L in P:
        return True
    elif SM(None, L.lower(), P.lower()).ratio()>0.5:
        return True
    else:
        return False

#============================================================================
#=======================       Extraer Problema        ======================
#============================================================================


def Idproblem(Paragraphs,F):

    iinit=0
    Search="problemajurídico"

    for i,P in enumerate(Paragraphs):
        if iinit!=0 and len(P.text)>3:
            if BoldT(P):
                return iinit-F,i-F,range(iinit-F,i-F,1)
            elif MayusT(P.text)  and len(P.text)<30:
                return iinit-F,i-F,range(iinit-F,i-F,1)
        if iinit==0 and"problem" in P.text and "jur" in P.text and len(P.text)< 34 and len(P.
    text)> 15:
            iinit=i
        elif iinit==0 and Search in P.text.lower() and len(P.text)< 34 and len(P.text)> 15:
            iinit=i
        elif iinit==0 and SM(None, Search, P.text.lower()).ratio()>0.54 and len(P.text)< 43
    and len(P.text)> 15:
            iinit=i
```

```
    if iinit!=0:
        return -1,-1,range(1,2,1)
    else:
        return -1,0,range(1,2,1)

# Final de Mayusculas ======================================================

def MayusT(Paragraph):
    PorcMayus=[True for Char in Paragraph if Char.isupper()]
    if len(PorcMayus)/len(Paragraph)>=0.75:
        return True
    else:
        False

# Final de NEgrilla ========================================================

def BoldT(Paragraph):
    PorcBold=[len(run.text) for run in Paragraph.runs if run.bold]
    Sumlen=np.sum(PorcBold)/len(Paragraph.text)
    if Sumlen>=0.75:
        return True
    else:
        return False

#==============================================================================
#=======================       Separacion Textos       ======================
#==============================================================================

#Lectura del documento y conversion a lista de parrafos ======================

def docReader(docs_file_name):
    Fallos=0
    Exitos=0
    FallosLimP=[]
    FallosLimT=[]
    print("====================================================================")

    for k,file in enumerate(docs_file_name):
        print("Documento: ",file)

        #Lectura del documento
        DOCFull=[]
        doc = docx.Document(Path+file)

        #Conversion a Lista de Strings
        for P in doc.paragraphs:
            DOCFull.append(P.text.lower())

        #Indice del parrafo final de la titulación
        F=IdTitulacion(doc.paragraphs)
```

```
    if F != -1:

        #Extraccion de la titulación en formato de DOCX
        Titulacion=DOCFull[:F]

        #Idices del problema juridico literal
        B,E,Range=Idproblem(doc.paragraphs,F)

        if E==-1:
            FallosLimP.append(file)

        if B != -1:

            Problema=DOCFull[B+1+F:E+F]

            if RealLen(Problema) and len(Problema)<len(DOCFull)*3/4:

                #Problema=DOCFull[B+1+F:E+F]
                print("Longitud Problema:",len(Problema))
                #Indices de los parrafos extraidos para la titulacion
                ITes=Finder(DOCFull[F:],Titulacion,Problema)

                if -1 not in ITes :

                    ITesO=list(OrderedDict.fromkeys(ITes))
                    print("Indesxs:",ITesO)
                    Tesis=[DOCFull[F+i] for i in ITesO]
                    ITes=sorted(set(ITes))

                    #Guardar archivos

                    SaveallCSV(file.strip(".docx"),Problema,Tesis,DOCFull[F:],FolderP+
Filename,FolderS,Range,ITes)

                    Exitos=Exitos+1
                else:
                    print("Fallo Tesis")
                    FallosLimT.append(file)
                    Fallos=Fallos+1
            else:
                print("Fallo Problema - Vacio")
                Fallos=Fallos+1
        else:
            print("Fallo Problema - No Finalizo")
            Fallos=Fallos+1
    else:
        print("Fallo Titulación")
        Fallos=Fallos+1
    print("Progreso:",k+1,"/",len(docs_file_name),"Documento: ",file,"\n")
    print("================================================================")
print("Fallos:",Fallos)
```

```
    print("Exitos:",Exitos)
    print("FallosP",FallosLimP)
    print("FallosT",FallosLimT)

#Longitud real problema ===============================================

def RealLen(Pro):
    for t in Pro:
        if len(t)>3:
            return True
    return False


#=======================================================================
#=========================       Guardar Textos        ================
#=======================================================================

# Eliminar Numeros iniciales en los parrafos y excluir la tesis y el problema

def Cleantext(DOC,Exclude):
    StrP=" "

    for i,P in enumerate(DOC):
        if i not in Exclude:
            NewP=P#.replace(".", " ")
            #NewP=re.sub(r"^(\s?)*\d+(\s?)\.+(\s?)\d*(\s?)\.*(\s?)"," ",P)
            StrP =StrP+NewP+"."
    return StrP


#Leer CSV, reamplazar o crear nueva fila con los datos ================

def SaveallCSV(Name,Problem,Thesis,Extra,PathFile,Year,Range1,Range2):


    #Crear textos ======================================================

    RangeT=list(Range1)+list(Range2)
    Exclude= sorted(set(RangeT))

    NewP=Cleantext(Problem,[-1])
    NewT=Cleantext(Thesis,[-1])
    NewJ=Cleantext(Extra,Exclude)

    #Nombre DOC, Carpeta/Año, Problema, Tesis, Restante ================

    Data=[Name,Year,NewP.lower(),NewT.lower(),NewJ.lower()]

    #Abrir CSV =========================================================

    df = pd.read_csv(PathFile)
```

```
    index = df.index
    Ilist = index[df["Documento"] == Name].tolist()

    #Registrar CSV ============================================================

    if len(Ilist)>=1:
        df.loc[Ilist[0]]=Data
    else:
        df.loc[len(df)]=Data

    #Guardar CSV ==============================================================

    df.to_csv(r'Dataset.csv',index=False)

#===============================================================================
#==============================       Ejecutar        ==========================
#===============================================================================

docReader(ToList())
```

Anexo 3. Código del programa utilizado para la identificación de las tesis y los problemas jurídicos dentro de las jurisprudencias.

```
#!/usr/bin/env python
# coding: utf-8
# # **Entrenamiento para la identificación de la tesis y el problema juridico en las
    sentencias de la sección primera del consejo de estado**
#
# ---
#
#
#
# # Librerias
# In[1]:
# Generales
import os
import re
import random
import string
import pickle
import numpy as np
import pandas as pd
import multiprocessing as mp
import matplotlib.pyplot as plt
#Sklearn - Entrenamiento
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, ConfusionMatrixDisplay
# NLTK
import nltk
from nltk import tokenize
from nltk.corpus import stopwords
from nltk.probability import FreqDist
from nltk.stem import SnowballStemmer
from nltk.metrics import ConfusionMatrix
# In[2]:
from spellchecker import SpellChecker
spanish = SpellChecker(language='es')
# # Lectura de datos
# In[3]:
#Direcciones
Path="C:\\Users\\willi\\OneDrive\\Escritorio\\Train\\"
```

```
PathSave=Path
# In[4]:
#Lectura dataset
df_s = pd.read_csv(Path+"Dataset.csv")
# In[5]:
df=df_s.sample(frac=1).reset_index(drop=True)
# In[6]:
#Dataset
df
# In[7]:
#Separación de columnas y etiqueta del texto
def Classification(df,Column,Num):
    data=[]
    dataNew=[]
    data1=list(df[Column])
    for P in data1:
        I=P.split(sep='.')
        for T in I:
            if len(T)>8:
                dataNew.append(T)
    data=[Num for e in range(len(dataNew))]
    df1=pd.DataFrame(dataNew,columns=['Texto'])
    df2=pd.DataFrame(data,columns=['Clasificacion'])
    df1['Clasificacion'] = df2['Clasificacion']
    return df1
# In[8]:
#Selección por columna y creación de la columna para clasificación
#Problemas=1; Tesis=2; Extra(Relleno)=0
df_3=Classification(df,'Extra',0)
df_1=Classification(df,'Problema',1)
df_2=Classification(df,'Tesis',2)
# In[9]:
print(len(df_3),len(df_2),len(df_1))
# # Pre-Procesamiento de datos
# In[10]:
syms=string.punctuation+'¿ ª¡`º  '
syms
# In[11]:
def Check_word(word):
    Ret=spanish.known([word])
    if len(Ret)>0:
        return True
    else:
        return False
# In[12]:
nltk.download('stopwords')
stop_words_es = set(stopwords.words('spanish'))
stemmer = SnowballStemmer('spanish')
def Clean_text(text):
    #Tabulaión textual
    text= text.replace("\t"," ")
```

```
    #Minusculas
    text = text.lower()
    #Puntuación y simbolos
    for sym in syms:
        text= text.replace(sym, " ")
    text=re.sub(r'[^\x00-\xFF]+',' ', text)#.replace("  ", " ").replace("ª"," ")
    #numeros
    text = re.sub(r'\d+',' ', text)
    #Espacios repetidos
    text = re.sub(r'\s+',' ', text)
    #Stop words y stem
    #text = " ".join([stemmer.stem(word) for word in text.split() if word not in stop_words_es
     and Check_word(word) and len(word)>3 and len(word)<13])

    text = " ".join([stemmer.stem(word) for word in text.split() if word not in stop_words_es
    and Check_word(word) and len(word)>3 and len(word)<13])

    return text
#
# In[13]:
def BalanceData(DFB):
    groups = DFB.groupby(DFB.Clasificacion)
    #groups.describe()
    DFB0=ms_df = groups.get_group(0)
    DFB1=ms_df = groups.get_group(1)
    DFB2=ms_df = groups.get_group(2)

    Bdf1 = DFB1.sample(n=int(len(DFB0)), replace=True)
    Bdf2 = DFB2.sample(n=int(len(DFB0)), replace=True)
    df = pd.concat([DFB0, Bdf1 ,Bdf2])
    df.reset_index(inplace=True, drop=True)

    return df['Texto'],df['Clasificacion']
# In[14]:
def SepData(DFB):
    groups = DFB.groupby(DFB.Clasificacion)
    #groups.describe()
    DFB0=ms_df = groups.get_group(0)
    DFB1=ms_df = groups.get_group(1)
    DFB2=ms_df = groups.get_group(2)

    Bdf1 = DFB1.sample(n=int(len(DFB1)*5), replace=False)
    Bdf2 = DFB2.sample(n=int(len(DFB1)*5), replace=False)
    Bdf0 = DFB0.sample(n=int(len(DFB1)*5), replace=False)
    df = pd.concat([Bdf0, Bdf1 ,Bdf2])
    df.reset_index(inplace=True, drop=True)

    return df['Texto'],df['Clasificacion']
# In[15]:
def SepData2(DFB):
```

```
    return DFB['Texto'],DFB['Clasificacion']
# In[16]:
#Pre procesamiento
df_1['Texto']=df_1['Texto'].apply(lambda x: Clean_text(x) )
df_2['Texto']=df_2['Texto'].apply(lambda x: Clean_text(x) )
df_3['Texto']=df_3['Texto'].apply(lambda x: Clean_text(x) )
# In[17]:
#union Dataframes
dfF = pd.concat([df_3, df_1 ,df_2])
dfF.reset_index(inplace=True, drop=True)
# In[18]:
X,Y = BalanceData(dfF)
# In[19]:
# Separación X e Y
#X=dfF['Texto']
#Y=dfF['Clasificacion']
X_train, X_test, y_train, y_test  = train_test_split(X, Y,train_size=0.75, random_state=42,
    stratify=Y)
# In[20]:
len(X_train)+len(X_test)
# # Datos de Entrenamiento
# # Entrenamiento
# In[21]:
import seaborn as sn
from sklearn.metrics import plot_confusion_matrix
def Results(y,y_predict):
    tn=['Extra','Problema','Tesis']
    #Reporte
    print(classification_report(y, y_predict, digits=4, target_names=tn))
    # Matriz
    cm=confusion_matrix(y, y_predict)
    ax = sn.heatmap(cm, annot=True, fmt='g',cmap='Blues');
    ax.xaxis.set_ticklabels(tn)
    ax.yaxis.set_ticklabels(tn)
    plt.show()
# In[22]:
NCore=int(-1)#int(mp.cpu_count())
NCore
# ## Baseline
# In[23]:
#Unigrama
vectorizer =CountVectorizer(ngram_range=(1, 1),min_df=10)
features_nd = vectorizer.fit_transform(X_train)
# In[24]:
vectorizer.get_feature_names()
# In[78]:
#Entrenar
log_model = LogisticRegression(max_iter=50000,class_weight='balanced',verbose=1,n_jobs=NCore).
    fit(X=features_nd, y=y_train)
# ### Train
# In[79]:
```

```
y_predT = log_model.predict(vectorizer.transform(X_train))
# In[80]:
Results(y_train, y_predT)
# ### Test
# In[81]:
y_pred = log_model.predict(vectorizer.transform(X_test))
# In[82]:
Results(y_test, y_pred)
# ## N-Gram
# In[30]:
from sklearn.naive_bayes import MultinomialNB
# In[31]:
#Bigrama y unigrama
vectorizer_NG =CountVectorizer(ngram_range=(1, 2),min_df=10)
features_nd_NG = vectorizer_NG.fit_transform(X_train)
# In[96]:
#Entrenar
log_model_NG = LogisticRegression(C=50,max_iter=50000,class_weight='balanced',verbose=1,n_jobs
    =NCore).fit(X=features_nd_NG, y=y_train)
#log_model_NG1 = MultinomialNB().fit(X=features_nd_NG, y=y_train)
# ## Train
# In[97]:
y_predT_NG = log_model_NG.predict(vectorizer_NG.transform(X_train))
# In[98]:
Results(y_train, y_predT_NG)
# ### Test
# In[99]:
y_pred_NG = log_model_NG.predict(vectorizer_NG.transform(X_test))
# In[100]:
Results(y_test, y_pred_NG)
# ## TF-IDF
# In[37]:
#vectorización
vectorizer_TF = TfidfVectorizer(min_df=2)
features_nd_TF = vectorizer_TF.fit_transform(X_train)
# In[38]:
#Entrenamiento
log_model_TF = LogisticRegression(max_iter=50000,class_weight='balanced',verbose=1,n_jobs=
    NCore).fit(X=features_nd_TF, y=y_train)
# ### Train
# In[39]:
y_predT_TF = log_model_TF.predict(vectorizer_TF.transform(X_train))
Results(y_train, y_predT_TF)
# ### Test
# In[40]:
y_pred_TF = log_model_TF.predict(vectorizer_TF.transform(X_test))
Results(y_test, y_pred_TF)
# ## Support Vector Machines (SVM)
# In[83]:
#Entrenamiento
```

```
log_model_svc = LinearSVC(max_iter=100, class_weight='balanced',verbose=1).fit(X=
    features_nd_NG, y=y_train)
# ### Train
# In[84]:
y_predT_svc = log_model_svc.predict(vectorizer_NG.transform(X_train))
Results(y_train, y_predT_svc)
# ### Test
# In[85]:
y_pred_svc = log_model_svc.predict(vectorizer_NG.transform(X_test))
Results(y_test, y_pred_svc)
# ## Pipe line model
# In[44]:
vectorizer_PL = TfidfVectorizer(ngram_range=(1, 2),min_df=2)
# In[45]:
pipeline = Pipeline([('vect',vectorizer_PL ),
                     ('chi',  SelectKBest(chi2, k=1200)),
                     ('clf', RandomForestClassifier(verbose=1,n_jobs=NCore))])
# In[46]:
#Entrenamiento
log_model_PL = pipeline.fit(X_train, y_train)
# ### Train
# In[47]:
y_predT_PL = log_model_PL.predict(X_train)
# In[48]:
Results(y_train, y_predT_PL)
# ### Test
# In[49]:
y_pred_PL = log_model_PL.predict(X_test)
# In[50]:
Results(y_test, y_pred_PL)
# # Guardar Modelos
# In[51]:
#Modelos
pickle.dump(log_model, open(PathSave+"Model_BL"+".pkl", "wb"))
pickle.dump(log_model_NG, open(PathSave+"Model_NG"+".pkl", "wb"))
pickle.dump(log_model_TF, open(PathSave+"Model_TF"+".pkl", "wb"))
pickle.dump(log_model_svc, open(PathSave+"Model_SVC"+".pkl", "wb"))
pickle.dump(log_model_PL, open(PathSave+"Model_PL"+".pkl", "wb"))
# In[52]:
#Vectorizadores
pickle.dump(vectorizer, open(PathSave+"vectorizer_BL"+".pkl", "wb"))
pickle.dump(vectorizer_NG, open(PathSave+"vectorizer_NG"+".pkl", "wb"))
pickle.dump(vectorizer_TF, open(PathSave+"vectorizer_TF"+".pkl", "wb"))
#
#
#
```

Anexo 4. Código del programa utilizado para la titulación de las jurisprudencias.

Código del programa utilizado para la titulación de las jurisprudencias, generado con ayuda de la herramienta de Qt Designer para el diseño de la interfaz.

```
import sys
import os
import re
import docx
import numpy as np
from os import listdir
from docx.shared import Pt
from os.path import isfile, join
from docx.enum.text import WD_BREAK
from docx.enum.text import WD_ALIGN_PARAGRAPH
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QFileDialog
from PyQt5 import QtCore, QtGui, QtWidgets
#Librerias ======================================================================
import string
import pickle
from difflib import SequenceMatcher as SM
#Sklearn - Modelos
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
# NLTK
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
nltk.download('stopwords')
#Documentos
# Funciones ======================================================================
# Busqueda de archivos
def ToList(ruta,ext):
    return [arch for arch in listdir(ruta) if (isfile(join(ruta, arch))) and ext in arch and "
    ~" not in arch]
#Pre Procesamiento
syms=string.punctuation+'¿ ª¡`°  '
nltk.download('stopwords')
stop_words_es = set(stopwords.words('spanish'))
stemmer = SnowballStemmer('spanish')
global Path
```

```
global Path2
Path="     "
Path2="     "
def Get_Path():
    global Path
    return Path
def Clean_text(text):
    #Tabulaión textual
    text= text.replace("\t"," ")
    #Minusculas
    text = text.lower()
    #Puntuación y simbolos
    for sym in syms:
        text= text.replace(sym, " ")
    text=re.sub(r'[^\x00-\xFF]+',' ', text)
    #numeros
    text = re.sub(r'\d+',' ', text)
    #Espacios repetidos
    text = re.sub(r'\s+',' ', text)
    #Stop words y stem
    text = " ".join([stemmer.stem(word) for word in text.split() if word not in stop_words_es
    and len(word)>3])
    return text
#Cargar Modelo
def LoadModel(Type,path):

    listP=["SVC","PL"]
    listL=["BL","NG","TF"]

    if Type in listP:

        NameModel="Model_"+Type+".pkl"
        Model=pickle.load(open(path+NameModel,'rb'))
        if Type == "PL":
            Model['clf'].set_params(verbose=False)
        print("Cargando modelo: "+NameModel+"...")
        return Model

    elif Type in listL:

        NameModel="Model_"+Type+".pkl"
        NameVector="vectorizer_"+Type+".pkl"
        Model_L=pickle.load(open(path+NameModel,'rb'))
        Vect=pickle.load(open(path+NameVector,'rb'))

        Model = Pipeline([('vect',Vect),('clf',Model_L)])

        print("Cargando modelo: "+NameModel+"...")
        return Model

    else:
```

```
        print("Fallo cargando modelo: "+NameModel+"...")

# Final de Mayusculas ========================================================
def MayusT(Paragraph):
    PorcMayus=[True for Char in Paragraph if Char.isupper()]
    if len(PorcMayus)/len(Paragraph)>=0.8:
        return True
    else:
        False

# Final de NEgrilla ==========================================================
def BoldT(Paragraph):
    PorcBold=[len(run.text) for run in Paragraph.runs if run.bold]
    Sumlen=np.sum(PorcBold)/len(Paragraph.text)
    if Sumlen>=0.8:
        return True
    else:
        return False
#Cargar Documento ============================================================
def LoadDocx(Num,path,listdocx,Path_docx):
    file=listdocx[int(Num)]
    doc = docx.Document(Path_docx+file)
    print("Cargando documento: "+file+"...")
    return doc
#Clasificar Parrafos =========================================================
def Clasification(Paragraphs,Model):
    Num0=0;Ix_Tesis=[];Ix_Probl=[];

    for i,P in enumerate(Paragraphs):
        #Predicción
        Prediction =Model.predict([Clean_text(P)])
        if Prediction==0:
            Num0=Num0+1
        elif Prediction==1:
            Ix_Probl.append(i)
        elif Prediction==2:
            Ix_Tesis.append(i)
        else:
            Num0=Num0+1

    #Resultados
    print("Resultados segun la clasificación de los algoritmos:")
    print("Extras=",Num0,"\t Problemas=",len(Ix_Probl),"\t Tesis=",len(Ix_Tesis))
    print("Tesis",Ix_Tesis)
    print("Problemas",Ix_Probl)

    return Ix_Tesis,Ix_Probl
# Identificar Inicio =========================================================
def IdTitulacion(P):
    String="consejo de estado"
```

```
    if String in P.text.lower() and len(P.text.lower())<20:
        return 1
    return -1
#Eliminar enumeraciones ============================================================
def DeleteNum(text):
    i=-1
    ipoint=text[:13].rfind(".")
    ig=text[:13].rfind("-")
    if ipoint>ig:
        i=ipoint
    else:
        i=ig
    if i>0:
        return text[i:]
    else:
        return text
# Eliminar incoherencias en el texto antes del guardado ======================
def clsave(text):

    text=DeleteNum(text)
    text=re.sub(r'\:+','. ', text)                                          #Doblepunto
    text=re.sub(r"^(\s?)*\d+(\s?)\.+(\s?)\d*(\s?)\.*(\s?)"," ",text)        #Numeros al
    comienzo
    text=re.sub(r"^(\s?)*\w(\s?)*\.+"," ",text)                             #Numeración
                            #
    text=re.sub(r"^(\s?)*\w(\s?)*\)+"," ",text)                             #Numeración
                            #
    text=re.sub(r'\s+',' ', text)                                           #Espacios
    Repetidos
    return text+"."
#Problema=============================================================================
def Idproblem(Paragraphs):

    iinit=0
    Search="problemajurídico"

    for i,P in enumerate(Paragraphs):
        if iinit!=0 and len(P.text)>3:
            if BoldT(P):
                TextP=""
                for NP in range(iinit+1,i,1):
                    if len(Paragraphs[NP].text)>5:
                        TextP=TextP+Paragraphs[NP].text
                return len(range(iinit+1,i,1)),TextP
            elif MayusT(P.text)  and len(P.text)<30:
                TextP=""
                for NP in range(iinit+1,i,1):
                    if len(Paragraphs[NP].text)>5:
                        TextP=TextP+Paragraphs[NP].text
                return len(range(iinit+1,i,1)),TextP
```

```
        if iinit==0 and"problem" in P.text and "jur" in P.text and len(P.text)< 34 and len(P.
    text)> 15:
            iinit=i
        elif iinit==0 and Search in P.text.lower() and len(P.text)< 34 and len(P.text)> 15:
            iinit=i
        elif iinit==0 and SM(None, Search, P.text.lower()).ratio()>0.54 and len(P.text)< 43
    and len(P.text)> 15:
            iinit=i


    if iinit!=0:
        return -1,"NN"
    else:
        return -1,"NN"
#estructuración de la lista de parrafos como un unico parrafo ================
def Iparagraphs(indexs,doc):
    Text=""
    for x,i in enumerate(indexs):
        if len(indexs)!=(x+1):
            if indexs[x]+3> indexs[x+1]:
                Text=Text+clsave(doc[i])+"[...] "
            else:
                Text=Text+clsave(doc[i])+""
        else:
            Text=Text+clsave(doc[i])+"."

    return Text
# Definir estilo de los parrafos de Tesis y de problema en el Doc ============
def setStyle(paragraph):
    paragraph.alignment = WD_ALIGN_PARAGRAPH.JUSTIFY
    for run in paragraph.runs:
        run.font.name = 'Arial'
        run.font.size = Pt(12)
        run.add_break(WD_BREAK.LINE)
        run.add_break(WD_BREAK.LINE)
# Guardar Parrafos en el Doc ==================================================
def SaveDoc(Name,DataT,DataP,document):
    #Abrir xls ================================================================
    global Path
    PathX=Path+"\Formated_docx_Files\\"
    #Leer Doc
    #Texto='Tesis:\n'+'\n'+DataT+"\n"+"\n"+"Problema:\n"+"\n"+DataP+"\n\n"       # Parrafos de
     Tesis y problemas
    #Texto=re.sub(r"(?<!\[)\.(\s?)+\.(?!\])",". ",Texto)                         # Eliminar
    puntuaciones repetidas

    DataT=re.sub(r"(?<!\[)\.(\s?)+\.(?!\])",". ",DataT)
    DataT=re.sub(r"\.\s\.",". ",DataT)
    DataP=re.sub(r"(?<!\[)\.(\s?)+\.(?!\])",". ",DataP)
    DataP=re.sub(r"\.\s\.",". ",DataP)
```

```
    setStyle(document.paragraphs[0].insert_paragraph_before(text="Tesis:"))    #Crear Parrafos
    setStyle(document.paragraphs[1].insert_paragraph_before(text=DataT))       #Crear Parrafos
    setStyle(document.paragraphs[2].insert_paragraph_before(text="Problema:")) #Crear Parrafos
    setStyle(document.paragraphs[3].insert_paragraph_before(text=DataP))       #Crear Parrafos
    document.save(PathX + Name + ".docx")                                       #Guardar
    dcoumento
def Identify(self,docs_files_name,listM,Path_models):
    ND=0
    Path_docx=Get_Path()
    for IdDoc in range(len(docs_files_name)):
        Name=str(docs_files_name[int(IdDoc)]).strip(".docx")                    #Nombre del
    docx
        Docx=LoadDocx(IdDoc,Path_docx,docs_files_name,Path_docx)                #Cargar docx
        #=========================================================================
        #Parrafos documentos
        P_docx=[]
        IT=False
        BE,Pro=Idproblem(Docx.paragraphs)
        for P in Docx.paragraphs:
            if not IT:
                IFTi=IdTitulacion(P)
                if IFTi > 0:
                    IT=True
            if IT and len(P.text)>3 and  not MayusT(P.text) and not BoldT(P):
                P_docx.append(P.text)
        print("Numero de Parrafos:", len(P_docx))
        #=========================================================================
        for model in listM:
            #=============================================================
            Model=LoadModel(model,Path_models)                                  #Cargar Modelo
            Tesis,Problema=Clasification(P_docx, Model)                         #Clasificar
    documento

            #=============================================================
            if len(Tesis)>0 and len(Problema)>0:
                if BE<3 or BE>15:
                    SaveDoc(Name,Iparagraphs(Tesis,P_docx),Iparagraphs(Problema,P_docx),Docx)
                    break
                if BE>2 and BE<16:
                    SaveDoc(Name,Iparagraphs(Tesis,P_docx),Pro,Docx)
                    break
        #Barra de progreso ====================================================
        progress=int(((ND+1)*100)/len(docs_files_name))                         #Progreso del
    EXE
        self.progressBar.setProperty("value", progress)                         #Asignar
    progreso
        ND=ND+1
        print("La App se ejecuto con exito !!!!!!!!!!")

#Carpetas ============================================================================
def IsFolderPath(Path_W):
```

```
    P1=Path_W +  "/Formated_docx_Files/"

    if not os.path.exists(P1):
        os.makedirs(P1)

def IsFolder():

    if not os.path.exists('Formated_docx_Files'):
        os.makedirs('Formated_docx_Files')
        print('Formated_docx_Files directory created!  \n')


#interface=================================================
#Codigo generado con la herramienta de Qt Designer==========
#===========================================================
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("Automatizador de Tesis y Problemas Juridicos")
        MainWindow.resize(598, 341)
        MainWindow.setWindowIcon(QIcon("CE.ico"))
        MainWindow.setWindowTitle("Automatización Tesis y Problemas Juridicos")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(170, 180, 241, 51))
        self.pushButton.setObjectName("pushButton")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 100, 211, 31))
        self.label.setObjectName("label")
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setGeometry(QtCore.QRect(30, 250, 561, 31))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")
        self.toolButton = QtWidgets.QToolButton(self.centralwidget)
        self.toolButton.setGeometry(QtCore.QRect(480, 100, 45, 31))
        self.toolButton.setObjectName("toolButton")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(110, 100, 360, 31))
        self.lineEdit.setObjectName("lineEdit")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(10, 10, 581, 21))
        font = QtGui.QFont()
        font.setFamily("century")
        font.setBold(True)
        font.setWeight(100)
        self.label_3.setFont(font)
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(10, 50, 581, 21))
        self.label_4.setObjectName("label_4")
```

```
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 598, 22))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        #Botones
        self.pushButton.clicked.connect(self.on_click)
        self.pushButton.setEnabled(False)
        self.toolButton.clicked.connect(self.SearchDOC)

        #Texto Path
        self.lineEdit.setEnabled(False)

    def retranslateUi(self, MainWindow):
        Mensaje="Bienvenido al software de detección de tesis y problemas juridicos de la
    sección primera del Consejo de Estado"
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Automatizador de Tesis y Problemas
     Juridicos"))
        self.pushButton.setText(_translate("MainWindow", "Buscar Tesis y Problemas"))
        self.label.setText(_translate("MainWindow", "Dir. de Origen:"))
        self.toolButton.setText(_translate("MainWindow", "..."))
        self.label_3.setText(_translate("MainWindow", "Automatizador de Tesis y Problemas
    Juridicos - Consejo de Estado"))
        self.label_4.setText(_translate("MainWindow", Mensaje))

    def on_click(self):
        #program RunTime
        #====================================================================
        listM=["NG","BL","PL"]                                              #Modelos a
    utilizar
        PathOrigin=Get_Path()                                               #Leer Path
        PathModels=".\\"

        docs_files_name=ToList(PathOrigin,".docx")
        self.pushButton.setEnabled(False)
        Identify(self,docs_files_name,listM,PathModels)
        self.pushButton.setEnabled(True)
        #====================================================================

    def SearchDOC(self):
        global Path
        global Path2

        try:
```

```
            Path =str( QFileDialog.getExistingDirectory(None, "Open a folder", 'C:/',
    QFileDialog.ShowDirsOnly))+"/"
            self.lineEdit.setText(Path)
            print(Path)
            if os.path.exists(Path) :
                IsFolderPath(Path)
                self.pushButton.setEnabled(True)
        except:
            Path=""
            self.pushButton.setEnabled(False)




if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```