UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Adaptación de Sentencias Judiciales a Lectura Fácil usando técnicas de Procesamiento de Lenguaje Natural

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Sánchez Sánchez, Alberto

Tutor/a: Hurtado Oliver, Lluis Felip

Cotutor/a: Segarra Soriano, Encarnación

Cotutor/a: Ahuir Esteve, Vicent

CURSO ACADÉMICO: 2023/2024

Resum

Per tal de cobrir les necessitats de persones amb diversitat intel·lectual sorgix la simplificació de text com una solució senzilla i eficaç que permet transcriure un text redactat amb llenguatge professional i formal, difícil de comprendre, en una versió més fàcil d'entendre. A partir del document original es genera un document en Lectura Fàcil, este concepte es defineix com l'adaptació d'un text a una versió que permet una lectura i una comprensió més senzilla del contingut. La Lectura Fàcil es dirigeix a totes les persones, en especial a les que tenen dificultats lectores.

Mitjançant l'ús de models de llenguatge, el projecte se centra en l'extracció de les idees principals de la part dispositiva de sentències judicials, el qual es planteja com un problema de classificació. D'altra banda, també s'extrau informació com la data, els curadors o la persona afectada, per a estes dades s'empra un model de Question Answering. Finalment, amb esta informació es crea un nou document en Lectura Fàcil.

Paraules clau: Lectura Fàcil, processament de llenguatge natural, grans models de llenguatge (LLM), classificació de text Zero-Shot, model Question Answering, adaptació de texts, diversitat intel·lectual

Resumen

Con el objetivo de satisfacer las necesidades de personas con diversidad intelectual surge la simplificación de texto como una solución sencilla y eficaz que permite transcribir un texto redactado con un lenguaje profesional y formal, difícil de comprender, en una versión más fácil de entender. A partir del documento original se genera un documento en Lectura Fácil, este concepto se define como la adaptación de un texto a una versión que permite una lectura y una comprensión más sencilla de su contenido. La Lectura Fácil se dirige a todas las personas, en especial a aquellas que tienen dificultades lectoras.

Mediante el uso de modelos de lenguaje, el proyecto se centra en la extracción de las ideas principales del fallo de sentencias judiciales el cual se plantea como un problema de clasificación. Por otra parte, también se extrae información como la fecha, los curadores o la persona afectada para la cual se emplea un modelo de Question Answering. Con la información extraída se crea un nuevo documento en Lectura Fácil.

Palabras clave: Lectura Fácil, procesamiento de lenguaje natural, grandes modelos de lenguaje (LLM), clasificación de texto Zero-Shot, modelo Question Answering, adaptación de textos, diversidad intelectual

Abstract

To meet the needs of people with different intellectual abilities, text simplification is used. It's a simple and effective solution that takes a formal and complex text, hard to understand, and turns it into an easier version by summarizing the main ideas and adjusting the language to what's needed. From the original document, an Easy-to-Read document is generated. This concept is defined as adapting a text to a version that allows for easier reading and undertanding of its content. Easy-to-Read is aimed at everyone, especially those who have reading difficulties.

Using language models, the project focuses on extracting key ideas from court judgments, treating it as a classification problem. It also extracts details like dates, guardians,

or the person involved using a question-answering model. This extracted information is used to create a new document in Easy-to-Read format.

Key words: Easy-to-Read, natural language processing, large language models (LLM), Zero-Shot text classification, Question Answering model, text adaptation, intellectual diversity

Índice general

Índice general v
Índice de figuras vii
Índice de tablas vii

1 Introducción 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 2
1.3 Interés social de la herramienta desarrollada . . . 2
1.4 Estructura de la memoria . . . 2
2 Contexto y marco teórico 5
2.1 Aplicación actual y aportaciones . . . 5
2.2 Lectura Fácil . . . 5
2.3 Fundación Espurna . . . 6
2.4 Clasificación de texto . . . 6
2.5 Question Answering (QA) . . . 6
2.6 Procesamiento del Lenguaje Natural (PLN) . . . 7
2.7 Transformers . . . 7
2.8 BERT . . . 8
2.9 XNLI . . . 9
3 Análisis del problema 11
3.1 Métricas de evaluación . . . 12
3.2 Descripción de los datos . . . 12
3.3 Modelos utilizados . . . 12
3.3.1 Modelo de clasificación Zero-Shot . . . 13
3.3.2 Modelo de QA . . . 13
3.4 Plan de trabajo . . . 14
4 Tecnologías utilizadas 15
4.1 HuggingFace . . . 15
4.2 Python . . . 15
4.3 Flask . . . 15
4.4 Transformers . . . 15
4.5 Pypdf . . . 16
4.6 HTML y Css . . . 16
4.7 JavaScript . . . 16
4.8 JSON . . . 16
4.9 Pandas . . . 16
4.10 Google CoLab . . . 16
5 Desarrollo del sistema 17
5.1 Punto de partida . . . 17
5.2 División de la sentencia . . . 18
5.3 Carga de los modelos . . . 23
5.4 Datos globales de la sentencia . . . 24

5.5 Clasificación 26
5.6 Entrenamiento Zero-Shot 28
5.6.1 Funcionamiento 28
5.6.2 Datos de entrenamiento 29
5.6.3 Librerías necesarias 29
5.6.4 Manejo de datos 30
5.6.5 Código de entrenamiento 30
5.7 Pictogramas 31
5.8 Visualización de los resultados 34
5.9 Otras aproximaciones y dificultades 36
5.9.1 Prompting 36
5.9.2 Dificultades en la fase de entrenamiento 37
5.9.3 Problemas con las cadenas de carateres 37
6 Evaluación de los resultados **39**
6.1 Evaluación QA 39
6.2 Evaluación clasificación 41
7 Conclusiones **43**
7.1 Relación con los estudios cursados 43
7.2 Trabajos Futuros 44

Apéndice

A ANEXOS **47**

Índice de figuras

2.1 Logo de Lectura Fácil 6
2.2 Logo de Espurna 6
2.3 Arquitectura Transformers 8
2.4 Ejemplo de Tokenización WordPiece 8

3.1 Funcionamiento del modelo Zero-Shot 13
3.2 Funcionamiento del modelo de Question Answering 13

5.1 Página web del trabajo anterior 17
5.2 Edición de los templates y las clases 18
5.3 División de la sentencia 21
5.4 División de la sentencia 22
5.5 Ejemplo de fallo con clases marcadas 26
5.6 JSON con los ejemplos de frases que se pueden seleccionar para cada clase 28
5.7 Ejemplo del primer conjunto de datos, fallo-hipótesis 29
5.8 Ejemplo del segundo conjunto de datos, frases-hipótesis 29
5.9 Pictograma Armas 32
5.10 Pictograma Testamento 32
5.11 Pictograma vehículos 32
5.12 Pictograma Estado civil 32
5.13 Pictograma Autocuidado 32
5.14 Pictograma Tratamientos 33
5.15 Pictograma Vida cotidiana 33
5.16 Pictograma Elecciones 33
5.17 Pictograma Residencia 33
5.18 Pictograma Centro de ingreso 33
5.19 Pictograma Patrimonio 34
5.20 Pictograma Dinero 34
5.21 Pictograma Poderes 34
5.22 Página web actualizada con los datos calculados 35
5.23 Ejemplo de los pictogramas en la página web 36
5.24 Ejemplo prompting 36

Índice de tablas

6.1 Evangelina ST UPV.pdf 39
6.2 Hortensia ST UPV.pdf 40
6.3 Precisión respuestas QA 40

6.4 Evaluación de métricas para diferentes umbrales (Sin Entrenamiento) . . . 41
6.5 Evaluación de métricas modelo entrenado con fallos 41
6.6 Evaluación de métricas modelo entrenado con frases 42

CAPÍTULO 1
Introducción

En la sociedad actual, una parte de la población tiene problemas de comprensión lectora, esto se debe a diversas causas. Por un lado puede deberse a causas transitorias: no conocer bien el idioma o una educación deficiente, por otro lado están las causas permanentes: dislexia, diversidad intelectual o TDAH. Ante este problema surge una solución, la Lectura Fácil [1], que en esencia es la adaptación de textos complejos en otros con un vocabulario más simple y una estructura más fácil de comprender.

Las sentencias judiciales de modificación de capacidad de obrar son aquellas en las que el juez decide quién se hará cargo de la curatela de la persona afectada, ya que considera que ésta no está capacitada para llevar una vida completamente independiente. Estos textos jurídicos se redactan con un lenguaje técnico y con una gramática y una sintaxis complejas que resultan muy difíciles de entender, más aún para estas personas que presentan dificultades lectoras. A partir de estas sentencias se ha creado una versión en Lectura Fácil, extrayendo las ideas principales de la sentencia y creando un nuevo documento que resume y simplifica el documento original sin perder información.

Esta funcionalidad es accesible en una página web a partir de la cual se puede elegir un documento, que se analiza y del cual se extraen los datos de mayor relevancia, para posteriormente crear el documento en Lectura Fácil. El resultado se puede visualizar en la web y además se puede descargar en formato word. De esta forma, se proporciona una herramienta para que la fundación Espurna o cualquier persona que lo necesite pueda generar textos en Lectura Fácil a partir de este tipo de sentencias, de forma rápida y sencilla.

1.1 Motivación

Una vez finalizada mi estancia Erasmus me puse en contacto con Encarna Segarra y Lluís Hurtado, profesores de SAR (Sistemas de Almacenamiento y Recuperación de Información) para preguntar si tenían algún tema sobre el que poder hacer mi TFG. Durante la carrera, esta asignatura ha sido una de las que más interesantes me ha parecido y me apetecía hacer el trabajo de fin de grado sobre algo relacionado con ello.

Me propusieron hacer este proyecto, el cual sería la continuación a dos trabajos anteriores, en este caso, con el objetivo de cambiar las técnicas que se utilizaban anteriormente y añadir nuevas funcionalidades. La idea me llamó mucho la atención, ya que, además de técnicas de procesamiento de lenguaje natural, también utilizaría técnicas de aprendizaje automático que es otra de las asignaturas que más me ha interesado de la carrera.

Por último, también me llamó la atención que fuese un proyecto con una necesidad real, que vaya a ayudar a gente que necesitase la versión en Lectura Fácil de las senten-

cias. Por este motivo, con Encarna Segarra y Lluís Hurtado, y junto a Vicent Ahuir, éste iba a ser mi tema de TFG.

1.2 Objetivos

El objetivo de este proyecto es la simplificación de sentencias judiciales en Lectura Fácil. En este proceso de simplificación distinguimos dos apartados: i) la extracción de información concreta que identifica la sentencia y ii) la simplificación del fallo de la sentencia.

A partir de un pdf o de un documento word, se analiza el texto y se realiza la extracción de la información relevante. Identificamos tres objetivos:

- Extracción de información relevante: Identificar datos concretos como la curadora, el juzgado, la fecha, la persona afectada, el tiempo máximo para apelar y el tiempo en el que se revisará la sentencia.
- Análisis y simplificación del fallo de la sentencia judicial:
  - Identificar y clasificar los elementos relevantes del fallo en diferentes temas.
  - Seleccionar oraciones en Lectura Fácil relativas a las clases previamente extraídas.
- De forma adicional, se incorporarán ayudas visuales como pictogramas o explicaciones de términos más complejos.

1.3 Interés social de la herramienta desarrollada

La aplicación de simplificación de sentencias de modificación de la capacidad de obrar a Lectura Fácil desarrollada en el proyecto tiene el objetivo de dotar a la Justicia y a organizaciones que se encarguen de la integración social de personas con diversidad intelectual, como es el caso concreto de la fundación Espurna, de una herramienta que les permita traducir las sentencias a Lectura Fácil de forma automática, precisa y al alcance de cualquiera.

De esta forma se facilita el acceso a documentos adaptados, en especial a personas con dificultades en la comprensión lectora.

1.4 Estructura de la memoria

1. Introducción: Breve introducción del proyecto, cuál es la motivación y qué objetivos se pretenden conseguir.
2. Contexto y marco teórico: Información sobre conceptos y soluciones actuales en los que se basa el proyecto y explicación detallada de la parte teórica de las tecnologías empleadas para el desarrollo de este proyecto.
3. Análisis del problema: Desde qué punto partimos y cómo vamos a abordar el problema. Aquí se analiza la estructura de las sentencias para plantear las posibles soluciones y los pasos a seguir. También se explica la información que utilizaremos para el desarrollo y el plan de trabajo, así como los modelos de lenguaje empleados.

4. Tecnologías utilizadas: Herramientas como aplicaciones, webs de consulta o librerías para el desarrollo del problema.

5. Desarrollo del sistema: Explicación detallada y ordenada de todos los pasos seguidos en el desarrollo de la solución.

6. Evaluación de los resultados: Métricas de los resultados obtenidos para cada aproximación y para cada uno de los problemas planteados.

7. Conclusiones: Revisión del proceso llevado a cabo y conclusiones que se derivan de los resultados obtenidos. Finalmente se sugieren diferentes mejoras que se podrían aportar en un futuro.

CAPÍTULO 2
Contexto y marco teórico

2.1 Aplicación actual y aportaciones

Se parte de una aplicación desarrollada en el marco del TFG del alumno Elías Esteve [2] y previamente en el TFG de Javier Meliá [3], dirigidos por los mismos tres directores. El funcionamiento de la aplicación consiste en un servidor que da acceso a una web donde se puede seleccionar un documento (pdf o Word). Internamente se analiza el texto del documento, se extrae el fallo de la sentencia y mediante análisis del lenguaje natural y con técnicas de similitud semántica, se seleccionan las clases que obtienen mayor puntuación en dicho análisis. Una vez obtenidos los resultados, se muestra en la web un prototipo del documento en Lectura Fácil. Hay funcionalidades extra que permiten descargar el documento en Word y editar las clases que han sido obtenidas en caso de que el usuario considere que ha habido algún error y quiera modificar el resultado.

Tras el cambio de normativa en 2021 [4], la redacción de las sentencias judiciales de modificación de la capacidad de obrar y su estructura cambian considerablemente de la regulación anterior. Este cambio hace necesaria la actualización de la aplicación que se había desarrollado anteriormente.

En la nueva versión de la aplicación desarrollada en este trabajo cambia la técnica de análisis del fallo para extraer las clases. En lugar de similitud semántica, hace uso de un modelo de clasificación de texto Zero-Shot con el que se probarán diferentes aproximaciones como modelos ajustados (*fine-tuned*). Este cambio de tecnología permite entender mejor las sentencias actuales, ya que los fallos son más ambiguos que con la normativa anterior y por ello, más difíciles de clasificar. Además, se añade la característica nueva de extracción de datos de la sentencia como la fecha, el juzgado o la persona que se encargará de la curatela del afectado. Por último, se incluirán elementos gráficos que ayuden a la comprensión del documento.

2.2 Lectura Fácil

Con el propósito de crear documentos que fueran fáciles de entender por cualquier persona, a finales de los 60 surgió en Suecia el concepto de Lectura Fácil [1]. En la actualidad, la Lectura Fácil se define oficialmente como un método que reúne un conjunto de pautas y recomendaciones para redactar, donde se tiene en cuenta el texto, el diseño y la maquetación del documento. También recoge normas de validación para comprobar si el documento es fácil de entender, con el objetivo de hacer accesible la información a las personas con dificultades de comprensión lectora.

Figura 2.1: Logo de Lectura Fácil

2.3 Fundación Espurna

La fundación Espurna [5] es una organización no gubernamental sin ánimo de lucro que se fundó en Gandía en 1996 con el objetivo de ayudar a la inserción socio-laboral de las personas con discapacidad intelectual. Su principal finalidad es proporcionar una formación personal que favorezca la integración en la sociedad y que mejore la calidad de vida de las personas con diversidad intelectual.

Figura 2.2: Logo de Espurna

2.4 Clasificación de texto

La clasificación de texto es una tarea fundamental en el Procesamiento del Lenguaje Natural (PLN), en la que se asignan etiquetas predefinidas a fragmentos de texto. Este campo ha experimentado un progreso significativo gracias a los avances en técnicas de aprendizaje automático.

Los métodos tradicionales para la clasificación de texto incluyen algoritmos de Machine Learning que utilizan diferentes representaciones del texto como por ejemplo bag-of-words o tf-idf, por otro lado, los avances en Deep Learning y las redes neuronales han revolucionado la clasificación de texto. Modelos como las Redes Neuronales Convolucionales (CNN) o Recurrentes (RNN) incluyendo variantes de Long-Short-Term Memory (LSTM) [6] han mejorado significativamente el rendimiento de diversas tareas de PLN. Uno de los avances más notables ha sido la introducción de modelos de Transformers como BERT [7], desarrollado por Google.

2.5 Question Answering (QA)

Las tareas de Question Answering (QA) implican el desarrollo de sistemas capaces de responder preguntas formuladas en lenguaje natural. Esta tarea es particularmente desafiante debido a la necesidad de comprender tanto la pregunta como el contexto en el que se encuentra la respuesta.

Inicialmente, los sistemas de QA se basaban en reglas y más adelante en técnicas de Machine Learning. Estos modelos permitían mapear preguntas a respuestas, pero eran modelos con muchas dificultades para manejar la variabilidad y la ambigüedad del lenguaje natural. La llegada del Deep Learning y los Transformers ha permitido que estos modelos no sólo comprendan una oración, sino que además puedan procesar textos de tamaño considerable para encontrar respuestas precisas.

Actualmente un ejemplo sería GPT-3 [8], desarrollado por Open-AI, el cual gracias a un enorme preentrenamiento es capaz de responder a preguntas complejas con mucha precisión y coherencia, incluso aborda tareas de generación de texto, resumen y traducción.

2.6 Procesamiento del Lenguaje Natural (PLN)

El PLN es un campo dentro de la lingüística y del Machine Learning que busca entender el lenguaje natural, no sólo palabras individuales, sino todo su contexto. Este campo permite realizar tareas como clasificar oraciones (si un correo es spam o no, si un tuit es negativo o positivo...), generar texto o extraer respuestas de un texto.

En el PLN, una de las tareas más importantes es traducir el texto en datos que los modelos puedan procesar, ya que éstos no pueden entender el texto como cadenas de caracteres y necesitan leer valores numéricos. Para ello primero se tokeniza la información en palabras, subpalabras, caracteres... según la necesidad del problema. Posteriormente estas divisiones se codifican en valores numéricos.

2.7 Transformers

Los Transformers son una arquitectura de modelos de aprendizaje profundo introducida por Vaswani et al. en 2017 [9] que ha revolucionado el campo del Procesamiento del Lenguaje Natural.

Muchos de estos Transformers son entrenados como modelos de lenguaje (BERT, GPT...), es decir, han sido entrenados con grandes cantidades de texto en crudo de manera auto-supervisada, de esta forma se consiguen modelos muy versátiles, pero que no son buenos realizando tareas específicas. Por este motivo, después de preentrenar al modelo es necesario realizar el ajuste (*fine-tuning*), esto permite entrenar al modelo para una tarea más específica. Con este segundo entrenamiento conseguimos que el conocimiento previo se transfiera (*transfer learning*) de forma que ahora se necesitan menos datos y menor coste de tiempo, recursos y ambiental para ajustar definitivamente el modelo.

La arquitectura de los Transformers se compone de dos partes principales como se puede observar en la Figura 2.3, el codificador (*encoder*) y el decodificador (*decoder*). El codificador toma la entrada y genera una representación interna, mientras que el decodificador toma esta representación y genera la salida. Cada una de estas partes se puede usar de forma independiente según la tarea que se desea hacer. Por último, una característica clave de los Transformers son las capas de atención (*attention*) como bien indica el título del artículo de Vaswani, *Attention is all you need*. Estas capas indican al modelo en qué partes de la oración se tiene que prestar especial atención, ya que para determinadas tareas como la traducción, una palabra será traducida teniendo en cuenta las palabras que le rodean, mientras que otras tareas como la predicción sólo podrán analizar las palabras anteriores a la que se quieren predecir.

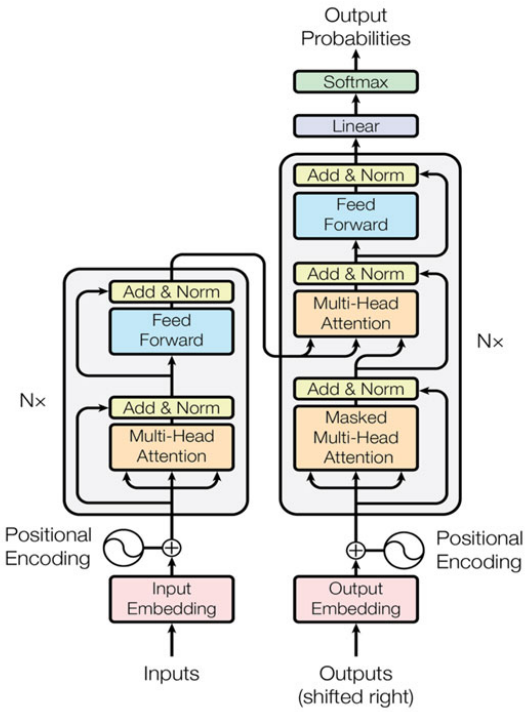


Figura 2.3: Arquitectura Transformers

2.8 BERT

BERT, *bidirectional encoder representation from transformers*, es un modelo de lenguaje preentrenado desarrollado por Devlin et al. [7]. Como su nombre indica, se distingue por su capacidad para considerar el contexto de forma bidireccional, es decir, tiene en cuenta tanto el contexto anterior como posterior de cada palabra de la secuencia.

Como se ha explicado anteriormente, cada modelo utiliza un tipo de tokenización diferente. BERT utiliza *WordPiece Tokenization* 2.4, donde se descomponen las palabras en subunidades más pequeñas (*tokens*), basadas en la frecuencia de aparición de secuencias de caracteres en el corpus de entrenamiento, de forma que se facilita la representación de palabras raras o desconocidas al subdividirlas en subpalabras más comunes.

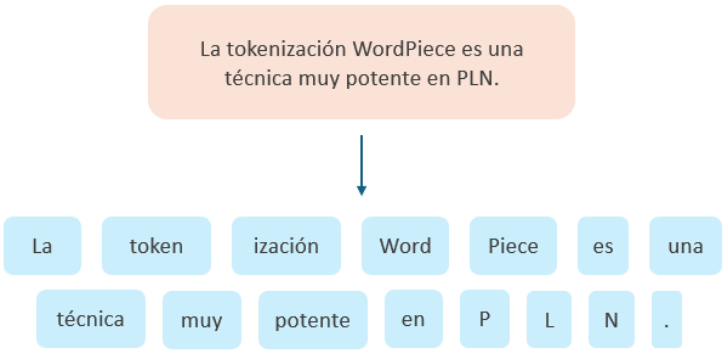


Figura 2.4: Ejemplo de Tokenización WordPiece

2.9 XNLI

La inferencia de lenguaje natural (*Natural Language Inference*, NLI) es una tarea imprescindible para el procesamiento de lenguaje natural que consiste en determinar la relación entre dos textos, la premisa y la hipótesis. Los textos se pueden relacionar de tres formas:

1. La premisa implica la hipótesis (*entailment*).
2. La hipótesis contradice a la premisa (*contradiction*).
3. La hipótesis no es necesariamente una conclusión lógica pero tampoco contradice a la premisa (*neutral*).

El proyecto XNLI [10], extiende la tarea de NLI a un contexto multilingüe y en él se proporciona un conjunto de datos y una metodología para evaluar la capacidad de los modelos de lenguaje para realizar inferencias de texto en múltiples idiomas. El corpus, como se ha explicado, contiene pares premisa-hipótesis traducidas a 15 idiomas, anotadas con una de las tres etiquetas de relación. El modelo empleado en este proyecto ha utilizado la porción en español de este dataset para su *fine-tuning*.

CAPÍTULO 3
Análisis del problema

Las sentencias judiciales contienen información concreta muy útil para el lector, sin embargo, el principal problema es que los textos jurídicos suelen estar formados por términos, conceptos y frases complejas que impiden encontrar esta información de forma rápida y clara.

Estos escritos se pueden dividir en secciones que mantienen una estructura común. La parte que contiene más información es el fallo, que se localiza al final del documento y en ocasiones, dentro de esta sección hay una subsección que contiene la información relativa a cómo impugnar o apelar la sentencia. Por otra parte, desde el inicio del documento hasta el fallo, la sentencia se puede dividir en introducción (o contexto), antecedentes de hecho y fundamentos de derecho, estas divisiones se pueden apreciar en el ejemplo de la Figura 5.3 y 5.4.

En los textos judiciales hay datos como la fecha o el juzgado que siguen una estructura parecida independientemente de quién lo escriba y se pueden localizar en la introducción o en las primeras líneas de los antecedentes de hecho. Además, en las sentencias de modificación de capacidad de obrar, que es el tipo concreto de texto jurídico con el que se va a trabajar en este proyecto, hay datos muy relevantes para las personas afectadas, como es la persona u organismo que será su curador según la sentencia y la propia persona que recibirá dicha ayuda, esta información puede localizarse en los primeros párrafos de la sentencia, ya sea en la introducción o al inicio de los antecedentes de hecho, o en los primeros párrafos de la parte del fallo. Además, la información relativa al método de impugnación de la sentencia y cada cuánto tiempo hay que revisarla, se encuentra en la parte del fallo, generalmente en los últimos párrafos de la sentencia.

Por último, la información necesaria para clasificar la sentencia se encuentra en el fallo, donde se puede obtener la decisión del juez respecto a los temas de los que el curador/a podrá hacerse cargo.

Este análisis permite centrarnos en partes muy concretas de la sentencia para buscar la información que queramos en cada caso y poder extraerla de la forma más eficiente posible, ya que los modelos trabajan mejor con extractos del texto reducidos, si el texto que les pasamos es muy extenso puede tardar demasiado tiempo en darnos una respuesta.

3.1 Métricas de evaluación

Para la evaluación de la calidad de los resultados utilizaremos las métricas de precisión, recall y puntuación F1.

Precisión: Compara la cantidad de valores que se han predicho correctamente (Verdaderos Positivos) entre el total de valores predichos.

$$\text{Precisión} = \frac{\textit{Clases Predichas Correctamente (VP)}}{\textit{Clases Predichas (VP + FP)}}$$

Recall: Compara la cantidad de valores predichos correctamente (VP) entre el total de valores que son correctos, es decir, todos los valores que habría que predecir en el caso ideal.

$$\text{Recall} = \frac{\textit{Clases Predichas Correctamente (VP)}}{\textit{Clases Reales (VP + FN)}}$$

Puntuación F1: Es la combinación de la precisión y el recall. Permite medir los resultados a partir de las dos métricas anteriores para tener una referencia más objetiva.

$$\text{F1} = 2 * \frac{\textit{Precisión * Recall}}{\textit{Precisión + Recall}}$$

3.2 Descripción de los datos

Para desarrollar el proyecto se han utilizado dos conjuntos diferentes de sentencias. El primero y más importante, compuesto por 14 sentencias judiciales de 2024 y 2023, es a partir del cual se han diseñado las preguntas que se realizan al modelo de Question Answering, ya que siguen la estructura y utilizan el léxico con el que se escriben las sentencias actualmente. Además, se ha hecho uso de un segundo conjunto con 50 sentencias anteriores a 2021, las cuales no siguen la estructura actual y emplean términos en desuso como "tutor". Este segundo conjunto se ha utilizado para el modelo de clasificación, ya que la sección del fallo no difiere en gran medida de los modelos actuales y ha permitido evaluar con mayor cantidad de ejemplos, así como entrenar el modelo Zero-Shot.

3.3 Modelos utilizados

Para este proyecto se han escogido dos modelos de HuggingFace [11]. En primer lugar, el modelo de clasificación Zero-Shot utilizado es *bert-base-spanish-wwm-cased-xnli* entrenado por *Recognai* [12]. En segundo lugar, el modelo de QA extractivo elegido para el proyecto es *mdeberta-v3-base-squad2* entrenado por *timpal0l* [13].

3.3.1. Modelo de clasificación Zero-Shot

La clasificación de textos Zero-Shot permite predecir una o más etiquetas que no habían sido vistas durante el entrenamiento del modelo. Este método puede considerarse un ejemplo de aprendizaje por transferencia (*transfer Learning*), es decir, utilizar un modelo entrenado para una aplicación distinta de aquella para la que se entrenó originalmente. Esto es realmente útil para tareas en las que la cantidad de datos etiquetados es reducida. De esta forma, podemos pasar como parámetros la porción de texto y un conjunto de posibles etiquetas, y éste puede predecir cuáles de estas etiquetas están más relacionadas con el texto. En la Figura 3.1 se puede ver de forma simplificada su funcionamiento.

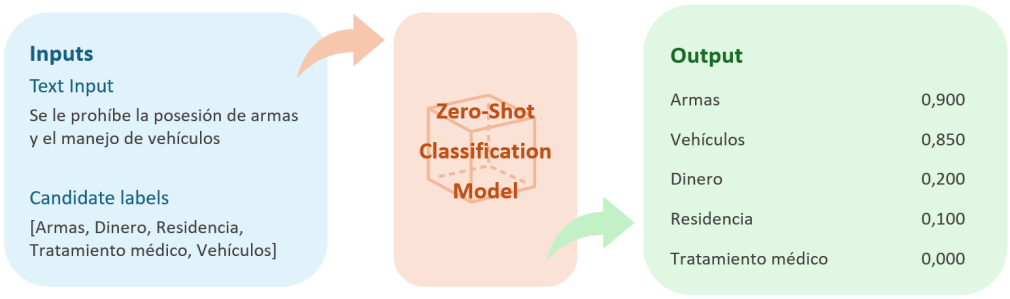


Figura 3.1: Funcionamiento del modelo Zero-Shot

El modelo utilizado en este proyecto, *bert-base-spanish-wwm-cased-xnli* [12], es una versión ajustada de la versión española del modelo BERT [14]. Para ello se ha utilizado la porción en español del dataset XNLI. Las siglas wwm (*Whole World Masking*) indican que para el entrenamiento se ha utilizado el método de enmascaramiento de palabras completas en lugar del enmascaramiento de subpalabras, este método permite mejorar la coherencia semántica.

3.3.2. Modelo de QA

Los modelos de QA pueden recibir como parámetro el contexto sobre el que se va a hacer la pregunta, aunque este contexto puede haberse proporcionado en la etapa de entrenamiento. En nuestro caso es un modelo de QA extractivo [13], esto significa que el modelo recibe el contexto personalizado y la pregunta que se desea formular, ya que estas preguntas serán sobre la sentencia en cuestión. En la Figura 3.2 se observa de forma simplificada el funcionamiento del modelo.

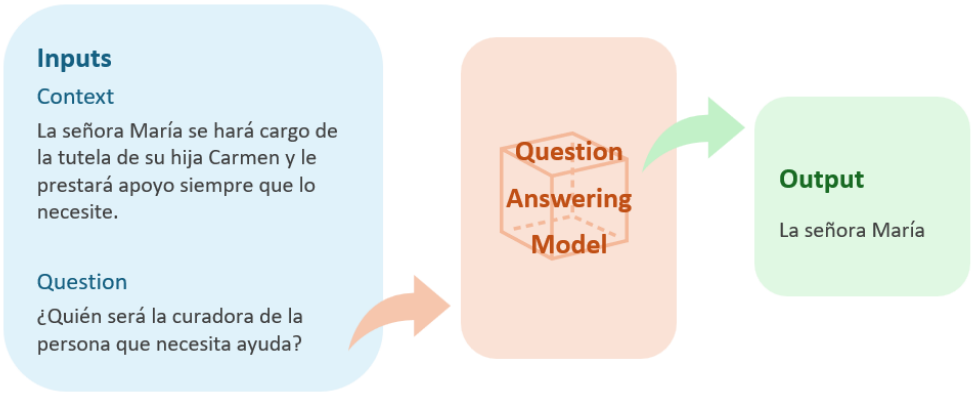


Figura 3.2: Funcionamiento del modelo de Question Answering

3.4 Plan de trabajo

Los pasos a seguir para este proyecto son los siguientes:

1. Analizar y entender los trabajos anteriores, principalmente el TFG de Elías Esteve. Familiarizarme con la estructura y el funcionamiento del programa para posteriormente poder modificarlo. 5.1
2. Estudiar aproximaciones con modelos de clasificación de texto y Question Answering y analizar diferentes implementaciones apropiadas para el problema.
3. Analizar los pdfs de las sentencias con los que se va a trabajar para encontrar similitudes en su estructura para posteriormente saber en qué partes de las sentencias se encuentra cada dato. 3
4. Mejorar la funcionalidad de lectura del pdf y estructuración de la información para poder analizarla de la forma más eficiente y añadir funciones que extraigan secciones del texto para que al aplicarlos a los modelos sea más rápido y eficaz. 5.2
5. Introducir la funcionalidad de búsqueda de datos (fecha, juzgado, curadora...) que utilizará el modelo de Question Ansering. 5.4
6. Introducir la funcionalidad de clasificación de texto. 5.5
7. Analizar diferentes aproximaciones para el modelo de clasificación de texto. *Fine-tuning* del modelo, probar diferentes modelos y evaluar los resultados. 5.9 y 6
8. Añadir funcionalidades gráficas extra: símbolos explicativos (pictogramas), subrayado del texto en el documento original y cuadros que definan términos. 5.7
9. Modificar el código HTML y JavaScript para incluir la información extraída de las sentencias y las características gráficas. 5.8

CAPÍTULO 4
Tecnologías utilizadas

4.1 HuggingFace

Hugging Face [11] es una plataforma *open-source* centrada en la inteligencia artificial y el Procesamiento del Lenguaje Natural (PLN). Es conocida por su biblioteca transformers, que proporciona herramientas y modelos preentrenados para tareas de PLN como traducción, clasificación de texto y generación de texto. Hugging Face facilita la implementación de modelos avanzados de Deep Learning, como BERT o GPT-3, permitiendo a los desarrolladores y científicos de datos aplicar técnicas de PLN de vanguardia en sus proyectos de manera sencilla y eficiente.

4.2 Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su simplicidad y legibilidad. Cuenta con una extensa biblioteca que la convierte en una herramienta versátil. Python se utiliza ampliamente en diversas áreas como desarrollo web, análisis de datos o inteligencia artificial.

4.3 Flask

Flask es un microframework de Python para el desarrollo de aplicaciones web. Es ligero y flexible, permitiendo a los desarrolladores crear aplicaciones web y servicios RESTful con facilidad.

4.4 Transformers

La librería Transformers de Hugging Face es una herramienta imprescindible para el Procesamiento del Lenguaje Natural (PLN). Facilita el uso de modelos de última generación como BERT o GPT para tareas como clasificación de texto, traducción o generación de texto. Simplifica la implementación de modelos preentrenados y su *fine-tuning* para aplicaciones específicas, haciendo accesible la tecnología de PLN avanzada a desarrolladores e investigadores.

4.5 Pypdf

La librería PyPDF es una herramienta diseñada para trabajar con archivos PDF. Permite realizar tareas como la extracción de texto, la manipulación de páginas o la fusión y división de documentos. Es útil para automatizar el procesamiento de documentos PDF.

4.6 HTML y Css

HTML y CSS son los fundamentos del diseño web. HTML permite crear la estructura de la página web mediante etiquetas, define encabezados, párrafos o imágenes. Css ayuda a crear el estilo de los elementos HTML definiendo colores, fuentes o disposiciones.

4.7 JavaScript

JavaScript es un lenguaje de programación de alto nivel utilizado principalmente para desarrollar aplicaciones web interactivas. Funciona del lado del cliente en el navegador, permitiendo crear funcionalidades como actualizaciones de contenido dinámico, animaciones o manejo de eventos y permite interactuar con HTML y Css para modificar y actualizar la estructura y el estilo de una página web en tiempo real.

4.8 JSON

JSON es un formato ligero de intercambio de datos, fácil de leer y escribir para los humanos, y fácil de parsear y generar para las máquinas. Se utiliza comúnmente para transmitir datos entre un servidor y una aplicación web como texto plano. JSON está basado en una estructura de pares clave-valor y soporta arrays y objetos anidados, lo que permite representar datos complejos de manera sencilla. Es independiente del lenguaje de programación, lo que lo hace ampliamente adoptado en diversas tecnologías y plataformas.

4.9 Pandas

Pandas es una biblioteca de Python para la manipulación y el análisis de datos. Ofrece estructuras de datos rápidas y flexibles, como DataFrame, que permiten manejar datos tabulares de manera eficiente.Permite cargar, procesar o exportar datos en diversos formatos como CSV, Excel y SQL.

4.10 Google CoLab

Google colaboratory es una plataforma gratuita de Google que permite a desarrolladores software ejecutar cuadernos jupyter en línea. Ofrece un entorno de ejecución en la nube con acceso a GPU y TPU, lo que facilita la realización de tareas intensivas de cálculo, como el entrenamiento de modelos de aprendizaje automático. Los usuarios pueden escribir y ejecutar código en Python, así como integrar otras herramientas populares de ciencia de datos. Además, Colab proporciona almacenamiento en Google Drive, permitiendo guardar y cargar archivos fácilmente.

CAPÍTULO 5
Desarrollo del sistema

5.1 Punto de partida

Elías Esteve, en su trabajo de fin de grado de 2023 [2], desarrolla la arquitectura del servidor y la página web que se han utilizado para la visualización de los resultados. Ahora explicaré brevemente el funcionamiento básico de ésta.

Se utiliza una arquitectura cliente-servidor. Mediante peticiones HTTP el cliente solicitará recursos al servidor y éste calculará los resultados y se los envía al cliente. La página web consiste inicialmente en una pantalla donde se podrá seleccionar un archivo pdf o Word a partir del que se quiere crear una versión en Lectura Fácil. Una vez el sistema procesa la sentencia, aparece la pantalla del resultado 5.1. Esta pantalla se divide en dos partes, el documento que se ha proporcionado aparece a la izquierda y la previsualización del documento en Lectura Fácil a la derecha.

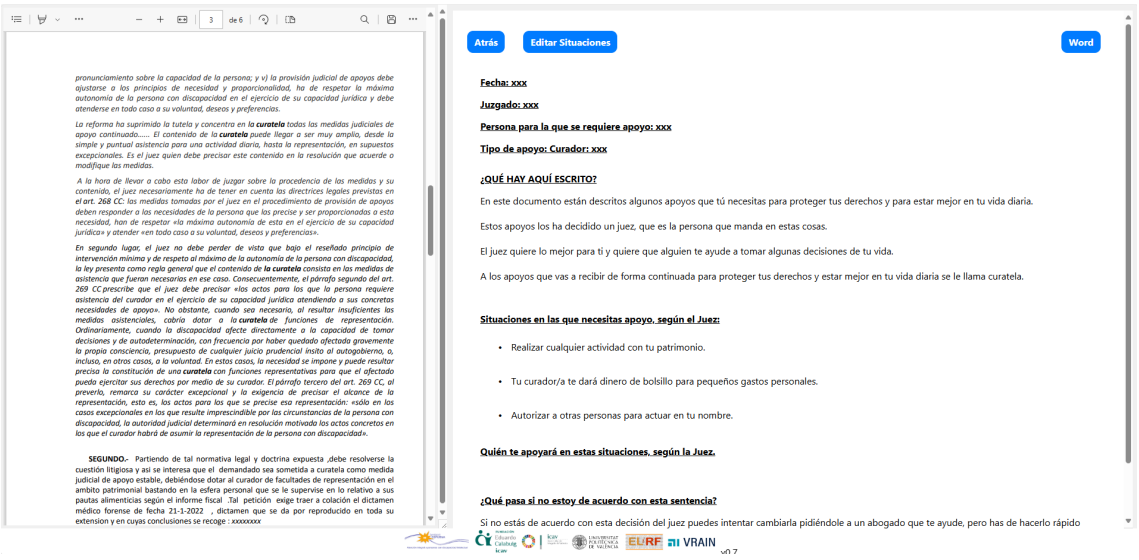


Figura 5.1: Página web del trabajo anterior

La web permite modificar las frases que han sido escogidas para cada clase de la sentencia, de forma que si el usuario quiere editar, añadir o eliminar la frase seleccionada para cada clase porque considera impreciso el resultado puede fácilmente editarlo como se puede observar en la Figura 5.2. Por último, la sentencia se puede descargar en formato Word gracias a la incorporación de un boton en la parte superior y su correspondiente funcionalidad.

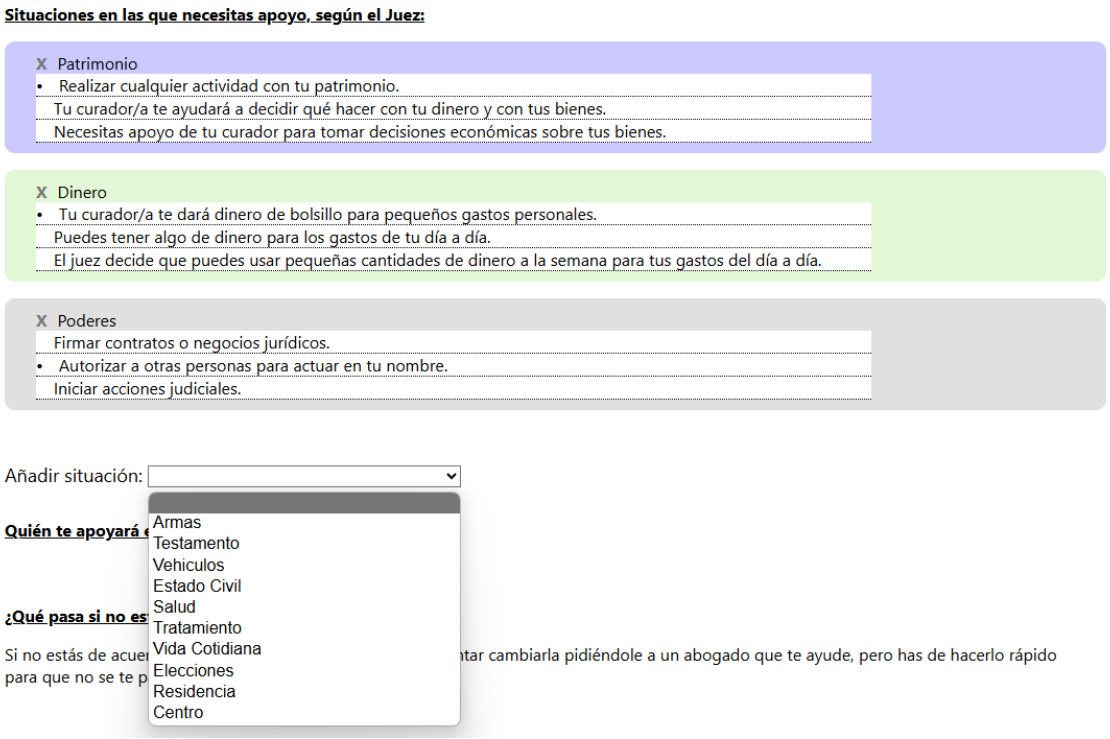


Figura 5.2: Edición de los templates y las clases

En los puntos que siguen a éste se explica el desarrollo de las características nuevas que se han añadido al trabajo anterior. Partiré explicando cómo el sistema lee y formatea el documento de entrada. Después se explica toda la lógica que utiliza modelos de lenguaje: la descarga de los modelos, el uso de éstos en Question Answering y clasificación, la aproximación que se ha escogido para entrenar los modelos y su funcionamiento. También se explicará cómo se han incorporado algunas ayudas gráficas y los cambios en la web para visualizar los resultados. Y acabaré resumiendo otras aproximaciones que se han considerado y las dificultades que he tenido durante el desarrollo del proyecto.

5.2 División de la sentencia

Para poder concretar las partes de la sentencia donde se extraerá la información, en la fase previa a la clasificación y extracción de información, se procesa todo el texto de la sentencia. En esta fase de procesamiento se da el formato más apropiado al texto, donde se crea un string con todo el texto en el que entre cada párrafo se introducen dos saltos de línea. En la Figura 5.1 se puede ver el método que lee el documento pdf y crea la variable con el texto.

Además, se crea un objeto de tipo diccionario 5.2, dónde se almacenará cada sección del texto y cada porción de la sentencia que tendrá relevancia en fases futuras.

```python
import pypdf
import re

def get_text_from_pdf(pdf):
    pdf_reader = pypdf.PdfReader(pdf)
    regex = r'\n\n+'
    text = ""

    for p in range(len(pdf_reader.pages)):
        page_text = pdf_reader.pages[p].extract_text(extraction_mode="layout")

        # Split por saltos de dos o mas lineas
        page_text = re.split(regex, page_text)

        # Saltos de linea unicos por espacios y unir todo por saltos dobles
        page_text = '\n\n'.join([e.replace('\n', ' ') for e in page_text])
        text += ' ' + page_text

    # Formateo espacios mutiples por espacios simples
    text = re.sub(r' +', ' ', text)

    # Obtengo el texto dividido correctamente por parrafos
    return text
```

Listing 5.1: Lectura del pdf

```python
    Sentencia = {
        "texto": "", ##toda la sentencia
        "contexto": "",  ## Parte de intrduccion del fallo
        "antecedentes de hecho": "",
        "fundamentos de derecho": "",
        "fallo": "",
        "modo impugancion": "" ## Si existe el apartado,

        ### Extractos del texto donde buscar la info de cada caracteristica
        "texto_curador": "",
        "texto_demandado": "",
        "texto_juzgado": "",
        "texto_fecha": "",
        "texto_impugnacion": "",
        "texto_revision": ""
    }
```

Listing 5.2: Diccionario con toda la información de la sentencia

Las sentencias se dividen en cinco partes: contexto, antecedentes de hecho, fundamentos de derecho, fallo y modo de impugnación. Las divisiones se pueden ver en el ejemplo de las imágenes 5.3 y 5.4. Esta estructura es común en la mayoría de los ejemplos de sentencias, pero en caso de no existir, o en caso de que el sistema no encuentre una de estas divisiones se reemplazará con regiones más amplias.

A partir de estas divisiones extraeremos porciones más concretas del texto 5.3 para encontrar los datos globales como quién será el curador/a, quién es la persona que recibe la ayuda, que relación familiar tienen entre ellos, la fecha y el juzgado de la sentencia e información relativa a los periodos de impugnación y revisión de la sentencia. En la sección 5.4 se detallan los criterios utilizados para seleccionar cada porción de texto para cada atributo y la pregunta que se formula en cada caso.

```python
import re

def texto_curador(sentencia):
    patron = r"[\n].*?(?:curador|curadora|curatela|tutor|tutora|tutela).*?[\n]"
    text = sentencia["contexto"] + sentencia["antecedentes de hecho"] + sentencia["fallo"]

    findings = re.findall(patron, text, flags=re.IGNORECASE)
    if findings:
        return " ".join(findings)
    return text

def texto_revision(sentencia):
    patron = r"[\n].*?revision.*?[\n]"
    fallo = extract_fallo(sentencia["texto"])
    findings = re.findall(patron, fallo, flags=re.IGNORECASE)
    if findings:
        return " ".join(findings)
    return fallo
```

Listing 5.3: Ejemplo de extracción de párrafos importantes para el curador y la revisión

Un ejemplo de sentencia con las cinco divisiones marcadas:

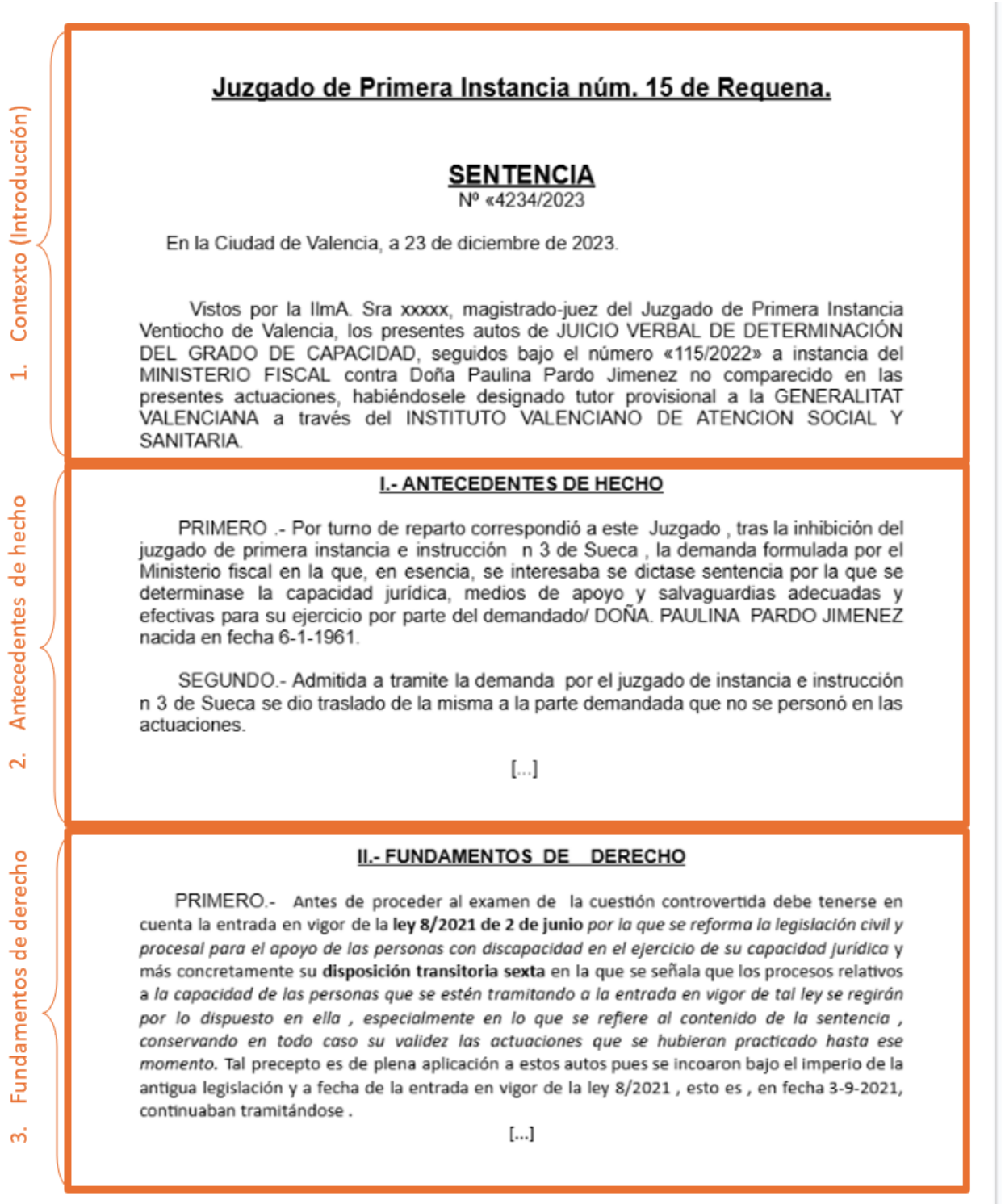


Figura 5.3: División de la sentencia

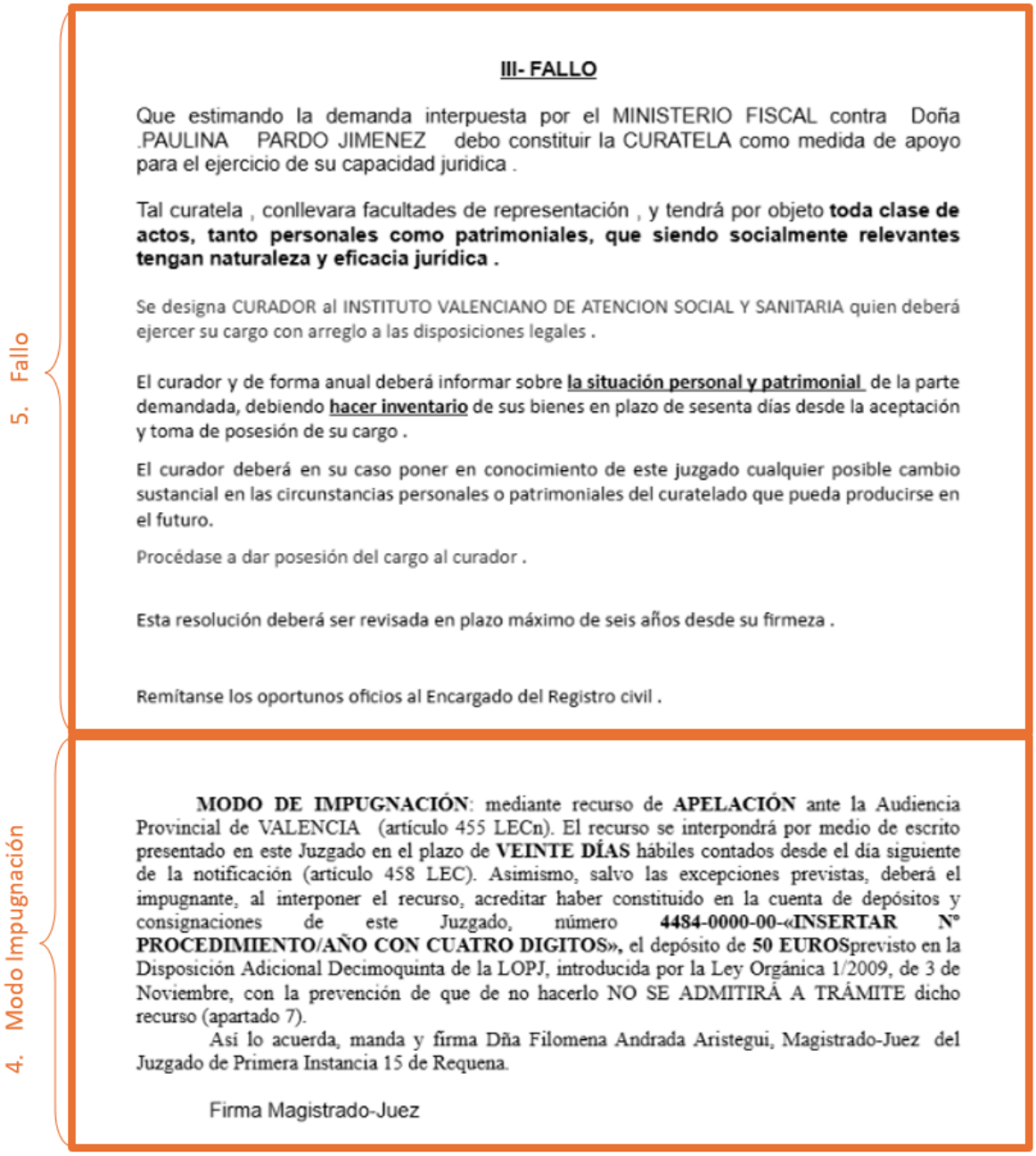


Figura 5.4: División de la sentencia

5.3 Carga de los modelos

En primer lugar, al inicio de la aplicación se cargarán los modelos que previamente hemos descargado de HuggingFace [13] [12]. Este proceso se realiza con tres sencillas líneas de código y se utilizan métodos de la librería *Transformers*, en la Figura 5.4 podemos ver el código del programa. La carga de los modelos difiere ligeramente según el tipo de tarea que queremos hacer, en este caso según sea un modelo de Question Answering o clasificación de texto Zero-Shot.

```
from transformers import pipeline
from transformers import AutoModelForQuestionAnswering
from transformers import AutoModelForSequenceClassification
from transformers import AutoTokenizer

# Rutas de los modelos guardados
qa_model_path = "./model/qa_model"
classification_model_path = "./model/classification_model"

# Cargar el pipeline de question-answering desde local
qa_model = AutoModelForQuestionAnswering.from_pretrained(qa_model_path)
qa_tokenizer = AutoTokenizer.from_pretrained(qa_model_path)
qa_pipeline = pipeline(
                        "question-answering",
                        model=qa_model,
                        tokenizer=qa_tokenizer
                    )

# Cargar el pipeline de clasificacion desde local
classification_model = AutoModelForSequenceClassification
                            .from_pretrained(classification_model_path)
classification_tokenizer = AutoTokenizer
                            .from_pretrained(classification_model_path)
classification_pipeline = pipeline(
                                "zero-shot-classification",
                                model=classification_model,
                                tokenizer=classification_tokenizer
                            )
```

Listing 5.4: Carga de modelos QA y clasificación de texto

Las clases *AutoModelFor - QuestionAnswering* o *SequenceClassification* proporcionan una interfaz para modelos de Question Answering o clasificación de secuencias respectivamente. Al utilizar el método *.from_pretained(path)*, se cargan los pesos y la configuración guardados en la ruta especificada de un modelo preentrenado (ya sea el base o con *fine-tuning*).

Como se explica en la sección 2.6 sobre procesamiento del lenguaje natural, cada modelo tiene asociado un tokenizador que traducirá cada secuencia para que pueda entenderla el modelo. De forma que *AutoTokenizer.from_pretained(path)* carga este componente que se asocia a nuestro modelo.

Por último, el método *pipeline* es uno de los más importantes de la librería *Transformers*. Es una función de alto nivel que permite agrupar un modelo y un tokenizador para realizar una tarea específica. Además, hay que especificar en el primer parámetro la tarea que queremos realizar, en nuestro caso 'question-answering' o 'zero_shot-classification', que generará el pipeline necesario para pasar los parámetros que cada tipo de tarea necesita.

5.4 Datos globales de la sentencia

Para extraer la información general de la sentencia se ha empleado un modelo de Question Answering [13], el cual recibe como parámetros de entrada un contexto y una pregunta (o dos en algunos casos). A continuación, se explica cada uno de los datos que se han extraído de la sentencia. Además, se define el criterio utilizado para extraer la porción de texto que se utiliza en cada caso, ya que es un contexto personalizado y específico para poder hacer más eficiente y eficaz la respuesta del modelo. También se detalla la pregunta que se formula al modelo para que la respuesta sea lo más precisa posible para cada atributo. En la Figura 5.5 se puede ver la función que recibe el contexto y la pregunta, y realiza la consulta al modelo de QA.

```
    def askQuestion(qa_model, question, texto):
        """
        Realiza una pregunta al modelo de Question Answering.

        Args:
            qa_model: El modelo de Question Answering.
            question: La pregunta que se desea hacer.
            texto: El contexto o texto relevante.

        Returns:
            Diccionario: {
                "answer": respuesta de la pregunta,
                "score": puntacion de la respuesta,
                "start": posicion de inicio de la respuesta,
                "end": posicion de fin de la respuesta
            }
        """
        output = qa_model(question=question, context=texto)
        return output
```

Listing 5.5: Ejemplo de pregunta al modelo de QA

Atributos de la sentencia con el contexto y la pregunta que pasarán al modelo de QA:

- **Fecha**
  - Contexto: Primera división de la sentencia, es decir el contexto de la sentencia.
  - Pregunta: "¿Cuál es la fecha en la que se presenta la sentencia?"
- **Curador/a**
  - Contexto: Unión de todos los párrafos de la sentencia dentro de las divisiones contexto, antecedentes de hecho y fallo, en los que aparece alguna de las siguientes palabras: curador/a, curatela, tutor/a, tutela, demandado, demandada/o. De esta forma se obtienen los párrafos más significativos donde se encuentra información relativa al curador.
  - Preguntas: En este caso se formulan dos preguntas diferentes y se selecciona aquella que mayor puntuación obtiene, esto se debe a que es diferente el caso en que el curador lo dictamine directamente el juez o sea asignado por una organización externa.
    1. "¿Quién será el tutor/curador del demandado y le prestará apoyo?"
    2. "¿Qué institución/organización asignará a un curador/tutor para que preste apoyo a la persona demandada?"

- **Demandado:** Persona que recibirá ayuda y a la que se aplica la sentencia.
  - Contexto: Similar al del curador, se añaden párrafos dónde aparezcan las palabras: demanda y demandado/a.
  - Preguntas: En este caso también se formulan dos preguntas diferentes y se escoge la que mayor puntuación obtiene, pero en este caso es debido a que las sentencias utilizan formas muy diferentes para referirse a esta persona:
    1. "¿Contra qué persona se ha interpuesto la demanda (quién es la demandada/o) por incapacidad?".
    2. "¿Quién necesita recibir medidas de apoyo de un curador?".
- **Revisión**
  - Contexto: Párrafos dentro del fallo o del modo de impugnación donde aparece la palabra revisión.
  - Pregunta: "¿En cuánto tiempo se revisará la sentencia?".
- **Impuganción**
  - Contexto: División modo de impugnación, y en caso de no existir se buscan los párrafos donde aparece alguna de las siguientes palabras: recurso de apelación, apelación, régimen de recursos o impugnar.
  - Pregunta: "¿Cuánto tiempo tiene el demandado para presentar una apelación o impugnar la sentencia?".
- **Juzgado:** Esta característica hace uso de expresiones regulares para encontrar el texto que coincida, éste tiene que empezar por la palabra 'juzgado', contener una de las cadenas 'instancia' o 'inst.' y terminará por coma o salto de línea.
  - Alternativa: En caso de que no se consiga encontrar texto que coincida (puede suceder porque se haya formateado mal o porque se haya usado otra estructura), se recurre a preguntar al modelo de QA.
    - Contexto: Contexto de la sentencia.
    - Pregunta: "¿Cuál es el nombre completo del juzgado?".
- **Parentesco:** Primero se analiza si el curador será una persona o una organización, para ello se analiza la cadena de texto con el resultado y en caso de que contenga palabras como 'ivass', 'fundación', 'organización'... no se analizará el parentesco puesto que será una organización. En caso de no contener una de estas palabras se pregunta al modelo:
  - Contexto: Todos los párrafos que contengan el nombre del curador o del demandado.
  - Pregunta: "Qué relación familiar tiene {curatela} con {demandado}". Cambiando los valores indicados.

5.5 Clasificación

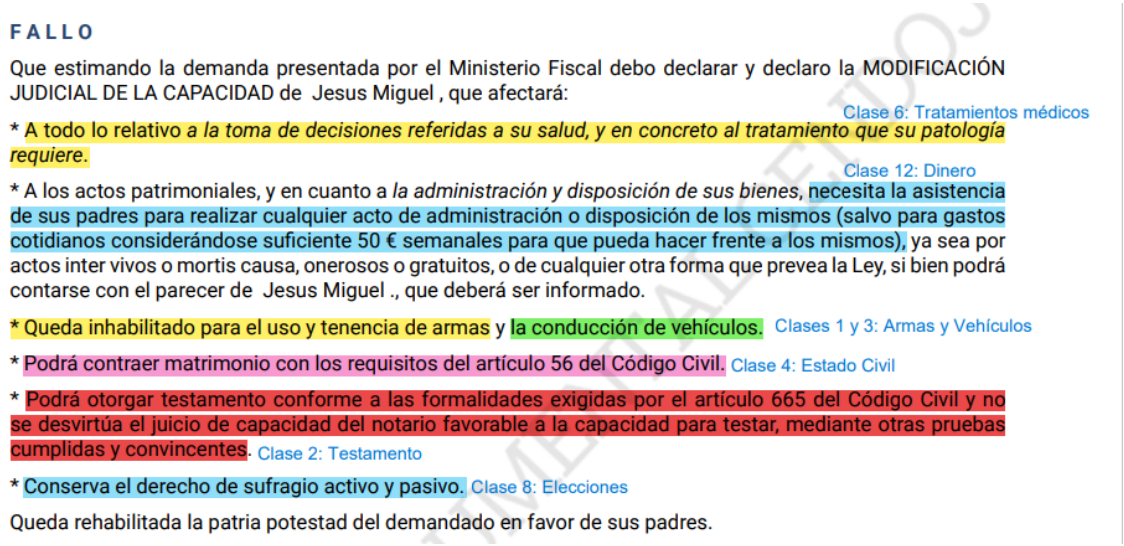
```
    # Se define un array con las 13 clases
    def getClasses(class_model, sentencia, labels=class_labels, umbral=0.35):
        result = class_model(sequences = sentencia["fallo"],
                             candidate_labels = labels,
                             multi_label = True
                            )

        # Conversion del resultado en un array de clases
        output = processResult(result, umbral)

        # Se elige una frase aleatoria para cada clase
        sol = {}
        for i in output:
            sol[i] = random.randrange(len(classes[str(i)]["template"]))
        return sol
    }
```

Listing 5.6: Función de clasificación

Para clasificación se emplea un modelo Zero-Shot de clasificación de texto [12] que permite etiquetar un texto sin necesidad de entrenamiento, únicamente pasando como parámetros el texto y las posibles clases.



FALLO

Que estimando la demanda presentada por el Ministerio Fiscal debo declarar y declaro la MODIFICACIÓN JUDICIAL DE LA CAPACIDAD de Jesus Miguel , que afectará:

Clase 6: Tratamientos médicos

* A todo lo relativo *a la toma de decisiones referidas a su salud, y en concreto al tratamiento que su patología requiere.*

Clase 12: Dinero

* A los actos patrimoniales, y en cuanto a *la administración y disposición de sus bienes*, necesita la asistencia de sus padres para realizar cualquier acto de administración o disposición de los mismos (salvo para gastos cotidianos considerándose suficiente 50 € semanales para que pueda hacer frente a los mismos), ya sea por actos inter vivos o mortis causa, onerosos o gratuitos, o de cualquier otra forma que prevea la Ley, si bien podrá contarse con el parecer de Jesus Miguel ., que deberá ser informado.

* Queda inhabilitado para el uso y tenencia de armas y la conducción de vehículos. Clases 1 y 3: Armas y Vehículos

* Podrá contraer matrimonio con los requisitos del artículo 56 del Código Civil. Clase 4: Estado Civil

* Podrá otorgar testamento conforme a las formalidades exigidas por el artículo 665 del Código Civil y no se desvirtúa el juicio de capacidad del notario favorable a la capacidad para testar, mediante otras pruebas cumplidas y convincentes. Clase 2: Testamento

* Conserva el derecho de sufragio activo y pasivo. Clase 8: Elecciones

Queda rehabilitada la patria potestad del demandado en favor de sus padres.

Figura 5.5: Ejemplo de fallo con clases marcadas

En las sentencias judiciales de modificación de la capacidad de obrar, la sección del fallo expone las situaciones en las que la persona afectada no podrá hacerse cargo de forma independiente, y por este motivo, la persona que se elija como curadora tendrá que ayudarle en todos estos puntos. Como vemos en el ejemplo de la Figura 5.5, el fallo expone que el afectado no podrá conducir vehículos, necesitará ayuda para manejar su dinero y tendrá que decidir conjuntamente con su curador decisiones importantes como vender/comprar una casa o contraer matrimonio con alguien (entre otras). Estos puntos son los que vamos a clasificar en una de las siguientes clases:

1. Armas.
2. Testamento.
3. Vehículos.

4. Estado Civil.

5. Autocuidado.

6. Tratamiento Médico

7. Vida Cotidiana.

8. Elecciones.

9. Residencia.

10. Centro de Ingreso.

11. Patrimonio.

12. Dinero.

13. Poderes Jurídicos.

Además, el modelo de clasificación recibirá como texto a clasificar únicamente la división de la sentencia con el fallo, ya que es donde se encuentra toda la información importante.

El resultado del modelo será una lista con las etiquetas ordenadas de más a menos probables, junto con la lista de valores de probabilidad que el modelo asocia a cada clase. Esta información nos permite crear un umbral a partir del cual seleccionar qué clases son positivas y cuales no tienen relación con el texto. En el código de la figura 5.6 se puede ver la función que nos devuelve las clases del texto.

A partir de esta lista de clases numéricas, se seleccionará una frase por cada clase que está en formato Lectura Fácil, que será la que se mostrará en el documento final. Esta frase se encuentra en un archivo JSON donde para cada clase se incluyen diferentes ejemplos (*template*) como podemos ver en la Figura 5.6 de los cuales se seleccionará uno aleatoriamente por cada clase.

```
{
  "1": {
    "name": "armas",
    "description": "posesión de armas",
    "template": [
      "El juez decide que no puedes tener ni utilizar armas de fuego.",
      "No puedes utilizar armas."
    ]
  },
  "2": {
    "name": "testamento",
    "description": "testamento",
    "template": [
      "Puedes hacer testamento, pero con condiciones.",
      "Si decides hacer testamento, tienes que hacerlo ante un notario."
    ]
  },
  "3": {
    "name": "vehiculos",
    "description": "conducción de vehículos",
    "template": [
      "El juez decide que no puedes conducir vehículos a motor, como el coche o la moto.",
      "No puedes utilizar vehículos."
    ]
  },
  "4": {
    "name": "estadocivil",
    "description": "matrimonio y estado civil",
    "template": [
      "También necesitas apoyo para cambiar tu estado civil."
    ]
  },

  [....]

}
```


Figura 5.6: JSON con los ejemplos de frases que se pueden seleccionar para cada clase

5.6 Entrenamiento Zero-Shot

Para mejorar los resultados experimentales de la clasificación se probarán dos procesos de *fine-tuning* del modelo base. Se seguirán dos aproximaciones que diferirán en los datos de entrenamiento.

5.6.1. Funcionamiento

El entremiento del modelo sigue la aproximación premisa-hipótesis que se ha introducido en el capítulo sobre XNLI 2.9, es decir, los datos de entrenmiento son pares de sentencias donde la primera es la premisa y la segunda es la hipótesis. El objetivo del modelo es determinar la relación entre estas dos sentencias, clasificándolas en una de las tres categorías: contradicción, implicación y neutro. El entrenamiento permite que el modelo entienda la relación semántica entre las oraciones y mejore los resultados para nuestra tarea específica.

5.6.2. Datos de entrenamiento

Para el entrenamiento se han creado dos conjuntos de datos distintos. En el primero de ellos las premisas son fallos completos de las sentencias cuyas hipótesis son los templates y las descripciones de las clases a las que pertenecen, ejemplo de la Figura 5.7. El segundo conjunto que podemos ver en el ejemplo de la figura 5.8, se ha elaborado extrayendo manualmente las frases más representativas de cada fallo y de igual modo, se han unido a las hipótesis de sus respectivas clases. Todas ellas etiquetadas como implicaciones.

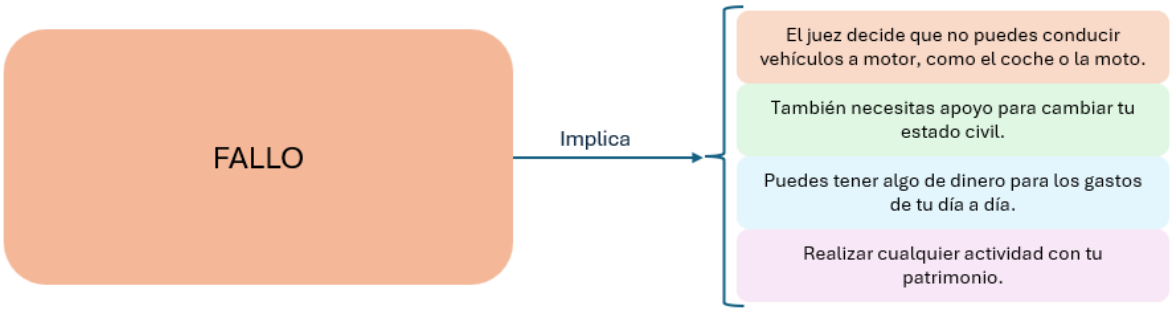


Figura 5.7: Ejemplo del primer conjunto de datos, fallo-hipótesis

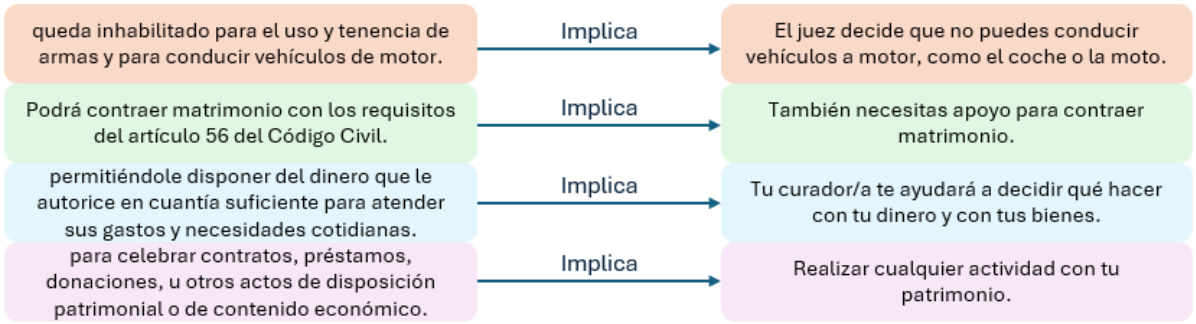


Figura 5.8: Ejemplo del segundo conjunto de datos, frases-hipótesis

5.6.3. Librerías necesarias

Para el entrenamiento del modelo es imprescindible instalar las librerías *transformers* y *datasets* las cuales nos proveerán de métodos para el manejo de datos y entrenamiento. En este caso se ha usado la versión específica 4.28 de la librería *transformers* ya que la versión más reciente generaba errores de ejecución y compatibilidad. También será necesario importar librerías como *pandas* o *torch*, y la función *train_test_split* para dividir los datos inicialmente.

5.6.4. Manejo de datos

En primer lugar se crean los *Datasets* necesarios para el entrenamiento. Se cargan los datos del documento xlsx y se dividen en datos de *training, test y validation*. Para poder trabajar con estos datos es necesario transformar los *Dataframes* de pandas en *Datasets* de HuggingFace, y con éstos crearemos un *DataDict* con las tres entradas.

```
df = pd.read_excel('/TFG/entrenamiento/fallos_hipotesis.xlsx')

train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
train_df, val_df = train_test_split(train_df, test_size=0.25, random_state=42)

# Convertir cada DataFrame en un Dataset de Hugging Face
train_dataset = Dataset.from_pandas(train_df)
val_dataset = Dataset.from_pandas(val_df)
test_dataset = Dataset.from_pandas(test_df)

# Crear un DatasetDict con las tres entradas
dataset = DatasetDict({
    'train': train_dataset,
    'validation': val_dataset,
    'test': test_dataset
})
```

Listing 5.7: Carga de los datos de entrenamiento

5.6.5. Código de entrenamiento

Para empezar el entrenamiento debemos crear un objeto *TrainingArguments* y otro *Training* que serán los que se encarguen de hacer el *fine-tuning*. Como vemos en el código de la Figura 5.8, los resultados se guardarán en el directorio especificado en *output_dir* y se guardará cada *save_steps* pasos. La estrategia de evaluación será *steps*, es decir, el modelo se evaluará cada cierto número de pasos en lugar de cada cierto número de *epochs*. El parámetro *no_cuda* nos permite elegir si queremos ejecutar el entrenamiento con GPU que será más rápido y para ello este valor tendra que ser **False**. Por último, establecemos *disable_tqdm* en **False** para ver el progreso del entrenamiento en tiempo de ejecución.

Una vez definidos los parámetros de entrenamiento, creamos el objeto *Trainer* al que le pasaremos estos argumentos, el modelo base sobre el que queremos hacer *fine-tuning* y los datos de entrenamiento y validación. Ejecutamos sobre este objeto el método *.train()* y el modelo empezará a entrenarse, es decir, irá mejorando los pesos específicos para que cada vez las predicciones sean más precisas. Este entrenamiento, como se ha explicado anteriormente, permite al modelo encontrar patrones gramaticales, sintácticos y léxicos entre la premisa y la hipótesis, de forma que más adelante, ante un caso distinto podrá predecir mejor las clases manteniendo el funcionamiento Zero-Shot.

```
training_args = TrainingArguments(
    output_dir='/content/drive/MyDrive/Colab_Results',

    do_train=True,  # Realizar entrenamiento
    do_eval=True,  # Realizar evaluacion
    evaluation_strategy="steps",  # Estrategia de evaluacion
    eval_steps=10,  # Pasos entre evaluaciones
    load_best_model_at_end=True,  # Cargar el mejor modelo al final
    metric_for_best_model="eval_accuracy",  # Metrica para el mejor modelo

    num_train_epochs=config["epochs"],
    per_device_train_batch_size=config["batch_size"],
    per_device_eval_batch_size=config["batch_size_eval"],

    warmup_steps=config["warmup_steps"],  # Pasos de calentamiento
    weight_decay=config["weight_decay"],  # Factor de decaimiento del peso
    learning_rate=config["learning_rate"],  # Tasa de aprendizaje

    logging_dir='./logs',  # Directorio para logs
    logging_steps=250,  # Pasos entre registros de log
    save_steps=10,  # Pasos entre guardados del modelo
    save_total_limit=5,  # Limite de checkpoints a guardar

    no_cuda=False,  # Usar GPUs si estan disponibles
    disable_tqdm=False,  # Mostrar barra de progreso
)

trainer = Trainer(
    model,
    tokenizer=tokenizer,
    args=training_args,                  # training arguments, defined above
    train_dataset=train_dataset,         # training dataset
    eval_dataset=valid_dataset,          # evaluation dataset
    compute_metrics=compute_metrics,
)

trainer.train()
```

Listing 5.8: Training Arguments

5.7 Pictogramas

Con el objetivo de que el usuario final tenga, además de una versión simplificada, una ayuda gráfica, se van a incorporar pictogramas a cada clase y a algunas secciones del documento para ayudar a la comprensión del mismo. Los pictogramas son pequeñas ilustraciones que permiten identificar visualmente el tema del que se está hablando, un buen ejemplo sería introducir la imagen de un coche cuando se está hablando de vehículos.

Las imágenes utilizadas se han descargado de la web *Global Symbols* [15] y a continuación se detalla qué pictograma se ha utilizado para cada clase:

- Clase 1: Posesión de armas

Figura 5.9: Pictograma Armas

- Clase 2: Testamento

Figura 5.10: Pictograma Testamento

- Clase 3: Manejo de vehículos

Figura 5.11: Pictograma vehículos

- Clase 4: Estado civil

Figura 5.12: Pictograma Estado civil

- Clase 5: Autocuidado

Figura 5.13: Pictograma Autocuidado

- Clase 6: Tratamientos médicos

Figura 5.14: Pictograma Tratamientos

- Clase 7: Vida cotidiana

Figura 5.15: Pictograma Vida cotidiana

- Clase 8: Elecciones

Figura 5.16: Pictograma Elecciones

- Clase 9: Residencia

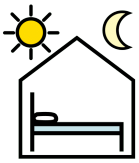


Figura 5.17: Pictograma Residencia

- Clase 10: Centro de ingreso

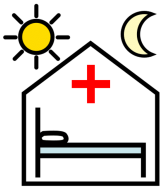


Figura 5.18: Pictograma Centro de ingreso

- Clase 11: Patrimonio

Figura 5.19: Pictograma Patrimonio

- Clase 12: Dinero

Figura 5.20: Pictograma Dinero

- Clase 13: Poderes jurídicos

Figura 5.21: Pictograma Poderes

5.8 Visualización de los resultados

Una vez extraída toda la información que vamos a necesitar de las sentencias es necesario modificar el código de la página web, tanto el HTML como el código JavaScript. Además se incluirán los pictogramas junto a la frase de cada clase.

El resultado se puede apreciar en la Figura 5.22. En la primera sección aparecen los cuatro datos más importantes de la sentencia: la fecha, el juzgado, el afectado y la persona u organización que se encargará de la curatela del afectado. Más adelante, se presentan las situaciones en las que el afectado necesitará apoyo, es decir, las clases que hemos calculado previamente. Visualmente esta parte de la web no cambia, únicamente cambia el proceso interno de cálculo. Después, se explica quién le prestará apoyo al afectado, en este apartado a parte de incluir al curador que se ha calculado, también se incluye el parentesco de éste con el afectado. Aquí se explica a la persona afectada quién es el curador y cómo podrá ayudarle. Al final del documento aparecen los datos relativos a cuanto tiempo tienen para apelar la resolución de la sentencia y cada cuanto tiempo deberá ser revisada.

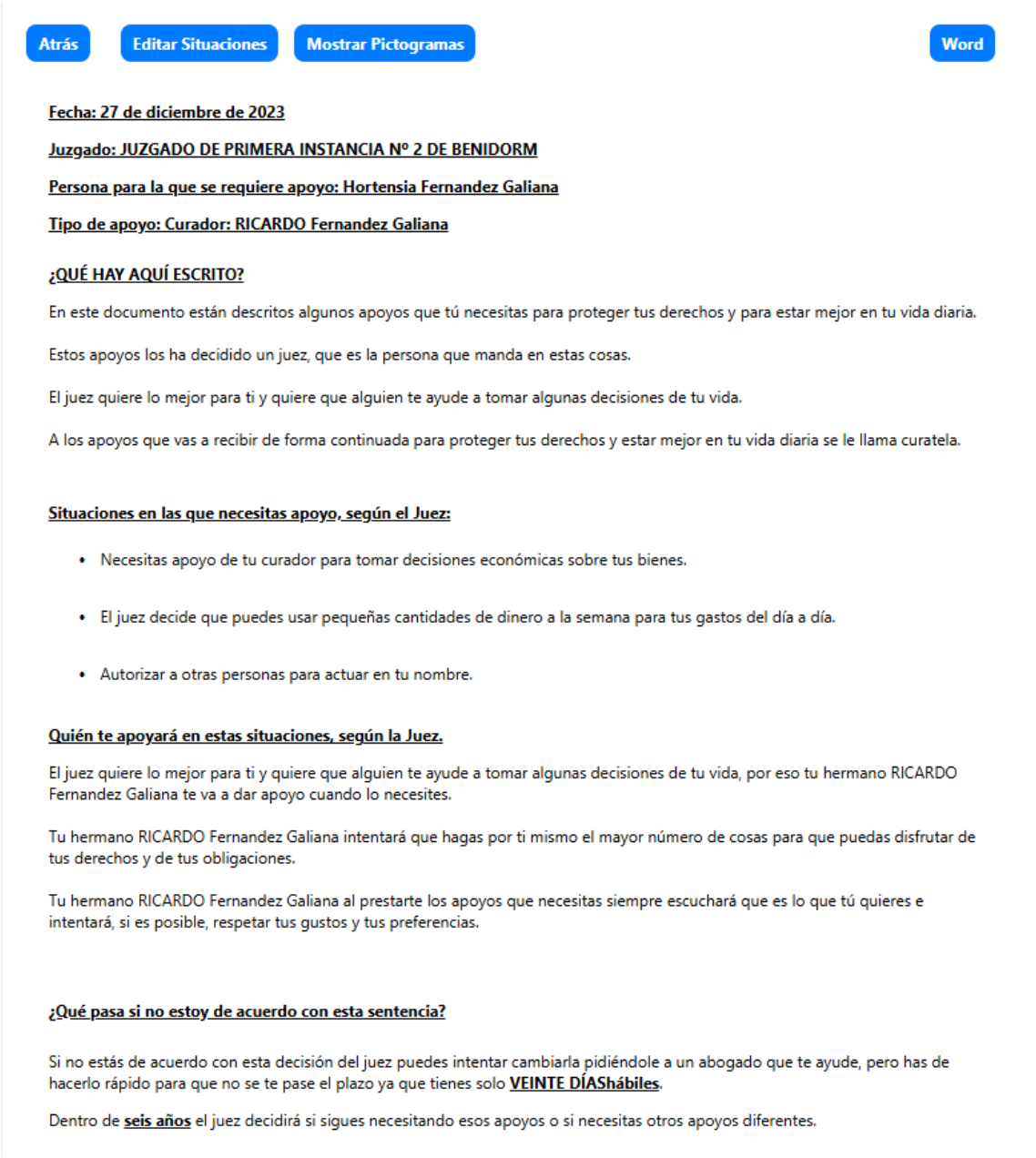


Figura 5.22: Página web actualizada con los datos calculados

Por otra parte se van a incluir los pictogramas presentados en la sección anterior 5.7. En la parte izquierda de cada frase seleccionada se añadirá un icono representativo de la clase. Esta característica será opcional, gracias al nuevo botón incorporado en la parte superior de la web, como se observa en la Figura 5.22, podrán ocultarse o mostrarse los pictogramas. El ejemplo anterior con pictogramas se visualizaría como la Figura 5.23

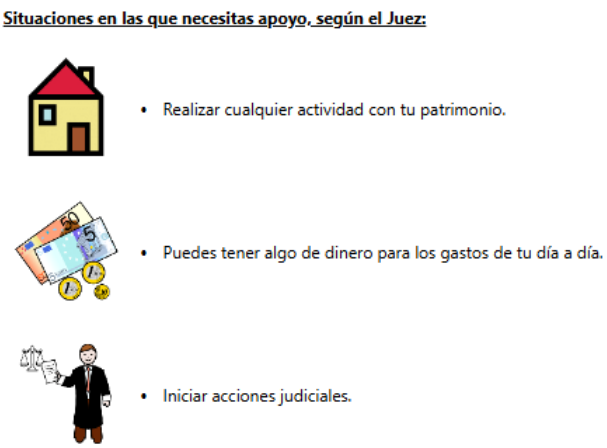


Figura 5.23: Ejemplo de los pictogramas en la página web

5.9 Otras aproximaciones y dificultades

Durante el desarrollo del proyecto he probado diferentes aproximaciones a los problemas que había que resolver, algunas de las cuales he acabado descartando. Además, en algunas fases del proyecto han surgido problemas que han conllevado cambios en la forma de proceder y he tenido que invertir, en algunos casos, mucho tiempo para encontrar una solución.

5.9.1. Prompting

Tanto para la extracción de características como para la clasificación, se probó una aproximación de *prompting*. Si bien el método utilizado de *fine-tuning* permite mejorar los resultados entrenando un modelo, el *prompting* permite introducir un ejemplo previo a la consulta para que el propio modelo tenga una referencia de cómo se quiere el resultado. En el ejemplo de la Figura 5.24 se generará la fecha con el formato indicado en el ejemplo previo de la consulta.

```
prompt = """Text: The first human went into space and orbited the Earth on April 12, 1961.
Date: 04/12/1961
Text: The first-ever televised presidential debate in the United States took place on September 28, 1
Date:"""
```

Figura 5.24: Ejemplo prompting

Esta solución finalmente no se ha contemplado ya que los tiempos de ejecución eran demasiado largos, principalmente debido a que la consulta era de muy larga extensión.

5.9.2. Dificultades en la fase de entrenamiento

Por otra parte, para el entrenamiento del modelo hubo varias complicaciones. En primer lugar, se intentó hacer el ajuste del modelo con una aproximación en la que se especificaban las clases directamente en la fase de entrenamiento, convirtiendo el modelo Zero-Shot en uno de clasificación de texto estándar. El principal inconveniente de esta solución es que las clases había que codificarlas para el entrenamiento, es decir, tenía que ser una lista numérica. Al entrenar directamente con clases numéricas se perdía el componente que relacionaba semánticamente la secuencia a clasificar con el nombre de la clase y por tanto no era una aproximación conveniente.

En esta fase, tanto para esta primera aproximación que fue descartada como para la aproximación de premisas e hipótesis que finalmente se tomó, hubo bastantes complicaciones que me hizo invertir mucho tiempo. Las versiones de las librerías necesarias para el entrenamiento, en especial *datasets* y *transformers* daban errores de compatibilidad. Inicialmente el error era un parámetro que había cambiado en la última actualización de la librería Transformers, de forma que al utilizarlo en unos métodos, en otros saltaba el error de que no era correcto. Este problema se solucionó encontrando una versión anterior en la que aún no saltaba este fallo.

5.9.3. Problemas con las cadenas de carateres

En las fases iniciales del proyecto, en la extracción del texto a partir del pdf surgió un problema con el formateo del texto. A la hora de convertirse a una cadena de caracteres se generaban espacios en blanco, saltos de línea o caracteres especiales que en el texto original no aparecían y por lo que el texto no se podía analizar correctamente. Este problema se pudo solucionar añadiendo el parámetro extraction_mode="layout"que se puede ver en la Figura 5.1.

Por último, en ocasiones la página web crasheaba sin motivo aparente. Encontrar el problema llevó un tiempo, hasta que me di cuenta que eran los strings que se pasaban como resultado en algunos campos. Si el juzgado, la fecha... contenían saltos de línea y hacían que al cargar esa información en JavaScript no se procesase correctamente la información. De manera que algunos de estos datos no aparecían en la web y además, las funcionalidades como 'atrás' o 'download' no estuvieran disponibles.

CAPÍTULO 6
Evaluación de los resultados

Los resultados obtenidos pueden ser erróneos, correctos o imprecisos... por eso es necesario evaluar la precisión de las dos funciones principales del proyecto.

Por un lado, para el modelo de QA se evaluará cada dato en conjunto y por separado, dependiendo de si es correcto o no. Por otro lado, el modelo de clasificación se evaluará con las métricas más comunes: precisión y recall. Es decir, se predecirán las clases y se puntuará la precisión en función de las clases correctas y erróneas en comparación a las reales. Para la clasificación se evaluará el modelo base con diferentes umbrales y las dos aproximación con *fine-tuning* del modelo base.

6.1 Evaluación QA

Ejemplos de evaluación del Question Answering:

Campo	**Dato real**	**Dato predicho**
Fecha	11/2/2021	27 octubre de 2021
Juzgado	Juzgado de primera instancia de Guadalajara nº 7	JUZGADO. 1ª INST. GUADALAJARA NÚM 7,
Demandado	Dña Evangelina	Dña. Evangelina
Curador	La fundación que designe la comisión de tutelas de Castilla La Mancha.	LA COMISIÓN DE TUTELAS DE CASTILLA LA MANCHA,
Impugnación	20 días	20 días
Revisión	6 años	SEIS años
Parentesco	null	null

Tabla 6.1: Evangelina ST UPV.pdf

Campo	Dato real	Dato predicho
Fecha	27/12/2023	27 de diciembre de 2023
Juzgado	De primera instancia nº2 de Benidorm.	JUZGADO DE PRIMERA INSTANCIA Nº 2 DE BENIDORM
Demandado	Hortensia Fernández Galiana	Hortensia Fernandez Galiana
Curador	Ricardo Fernández Galiana.	El hermano de la afectada
Impugnación	20 días	VEINTE DÍAS hábiles
Revisión	6 años	seis años
Parentesco	hermano	hermano

Tabla 6.2: Hortensia ST UPV.pdf

Como se puede observar, los resultados no coinciden en mayúsculas, dígitos o signos de puntuación, esto se debe a que el texto se extrae directamente del texto y no se puede formatear correctamente dada la infinidad de posibilidades que puede tener una cadena de caracteres. De este modo, se considera correcto todo campo que sea equivalente y en los ejemplos anteriores sólo consideraríamos incorrecto la fecha en la Tabla 6.1 o el curador en la Tabla 6.2.

Los resultados desglosados por campo y en conjunto se pueden ver a continuación en la Tabla 6.3. Para esta evaluación sólo se han tenido en cuenta las 14 sentencias más actuales, ya que las anteriores a 2021, debido al cambio de normativa no sigue la misma estructura ni utiliza el mismo léxico, por este motivo, los datos no se predicen correctamente. Para el caso concreto del parentesco, sólo 10 de las sentencias tienen como curador a una persona, por ello, para este atributo sólo se evalúan estas 10 muestras.

Campo	Precisión
Fecha	0.93
Juzgado	1
Curatela	0.85
Demandado	0.93
Impugnación	0.93
Revisión	1
Parentesco	0.8
Precisión general	**0.92**

Tabla 6.3: Precisión respuestas QA

Los resultados obtenidos que se pueden ver en la Tabla 6.3 para la evaluación del modelo de Question Answering son muy positivos. Por un lado el juzgado y el periodo de revisión los ha predicho bien en todos los casos ya que es información que se presenta de la misma forma en casi todas las sentencias. Por el contrario, la curatela junto con el parentesco son los campos en los que mayor error da el modelo, esto se debe a que en las sentencias es uno de los atributos que más difiere en la forma de presentarlo, y por tanto, el modelo lo predice peor porque no encuentra palabras con alta similitud semántica respecto a la pregunta, es decir, no entiende bien quién es el curador.

6.2 Evaluación clasificación

Para el entrenamiento y evaluación de las predicciones del modelo de clasificación Zero-Shot se ha utilizado el primer conjunto de 14 sentencias junto con el segundo conjunto de 50 sentecias, es decir, todas las sentencias disponibles ya que los fallos no cambian su estructura de un año para otro. Para esta evaluación se han comparado tres modelos: sin entrenamiento, con entrenamiento de pares fallo-hipótesis y con entrenamiento de pares frase-hipótesis. Para el entrenamiento y evaluación del segundo y tercer modelo se han dividido los 64 fallos aleatoriamente, un 80 % para entrenamiento y un 20 % para la evaluación.

Umbral	Precisión	Recall	F1
0.05	0.847	0.611	0.71
0.1	**0.721**	**0.728**	**0.725**
0.15	0.622	0.74	0.67
0.2	0.532	0.729	0.615
0.25	0.428	0.718	0.53

Tabla 6.4: Evaluación de métricas para diferentes umbrales (Sin Entrenamiento)

En la Tabla 6.4 podemos observar como el mejor umbral es 0.1, un umbral muy bajo para seleccionar las clases a las que pertenece el fallo. La precisión es muy alta para umbrales más bajos ya que se filtran menos clases de manera que la predicción devuelve mayor número de clases, lo que supone acertar más clases de las que son correctas. Sin embargo, el *recall* funciona de manera inversa, aumenta a medida que el umbral es mayor, esto sucede porque al acotar la cantidad de clases, las que resultan finalmente escogidas son mayoritariamente correctas.

Umbral	Precisión	Recall	F1
0.1	0.871	0.596	0.708
0.15	0.829	0.633	0.718
0.2	0.811	0.67	0.734
0.25	0.802	0.687	0.74
0.3	0.784	0.704	0.742
0.35	**0.766**	**0.721**	**0.743**
0.4	0.742	0.72	0.731
0.45	0.736	0.734	0.735

Tabla 6.5: Evaluación de métricas modelo entrenado con fallos

Por otra parte, la evaluación del modelo previamente entrenado con fallos como premisas mejora ligeramente los resultados. No vemos una gran mejora ya que la cantidad de datos de la que disponemos es escasa, además, los fallos de las sentencias son muy ambiguos, lo que en ocasiones clasifican como 'patrimonio' otras lo clasifican como 'dinero' o 'poderes' independientemente de lo que diga explícitamente el texto. En esta ocasión, el mejor resultado sería para el umbral 0.35.

Umbral	Precisión	Recall	F1
0.1	0.596	0.881	0.711
0.15	0.633	0.839	0.722
0.2	0.670	0.821	0.738
0.25	0.687	0.812	0.744
0.3	0.704	0.793	0.746
0.35	**0.721**	**0.775**	**0.747**
0.4	0.720	0.751	0.735
0.45	0.734	0.745	0.739
0.5	0.739	0.715	0.727
0.55	0.750	0.700	0.724

Tabla 6.6: Evaluación de métricas modelo entrenado con frases

Por último, el modelo entrenado con extractos de los fallos relacionados con cada clase nos da los resultados de la Tabla 6.6. De igual modo, la mejora respecto a los modelos anteriores es muy leve.

CAPÍTULO 7
Conclusiones

Una vez finalizado el proyecto puedo decir que se han completado la mayoría de los objetivos planteados inicialmente. En primer lugar, la clasificación del fallo con el uso del modelo Zero-Shot ha sido una buena solución dado el escaso volumen de muestras del que se disponía y de la ambigüedad de los textos a causa de la nueva normativa de las sentencias jurídicas. Y, en segundo lugar, la extracción de características mediante el modelo de Question Answering también ha sido una solución muy acertada para la tarea, en este caso con porcentajes de acierto muy altos. El tercer punto de los objetivos principales del trabajo, las ayudas gráficas, se han completado parcialmente. Se ha podido incorporar los pictogramas al documento, mientras que los cuadros explicativos, por cuestiones de tiempo no se han podido añadir, siendo ésta la principal característica que queda para una futura actualización del proyecto.

En el desarrollo del proyecto han surgido dificultades que han supuesto mucho tiempo como se comenta en el capítulo 5.9, en especial el entrenamiento del modelo. También ha requerido mucho tiempo investigar y familiarizarme con tecnologías con las que no había trabajado antes. Gracias a esta investigación he descubierto un campo muy extenso que es la es la inteligencia artificial y todas sus posibilidades y me ha hecho darme cuenta de que quiero enfocar mi carrera hacia esta rama de la informática.

Por último, este proyecto me ha permitido colaborar en la mejora de una herramienta que ayudará a mejorar la calidad de vida de algunas personas, lo cual es un añadido a un trabajo que he disfrutado y del que me siento orgulloso de haber desarrollado satisfactoriamente.

7.1 Relación con los estudios cursados

El primer cuatrimestre del cuarto año de carrera cursé una asignatura en mi estancia de Erasmus en Milán llamada Deep Learning and Neural Networks. Desde mi punto de vista es la asignatura más relacionada con la tecnología que se utiliza en este trabajo que son los Grandes Modelos de Lenguaje, en esta asignatura aprendí las bases del Aprendizaje Automático y, aunque no llegué a ver este tipo de modelos, aprendí a trabajar con otro tipo de modelos similares a estos y a entrenarlos.

Por otra parte, la asignatura que imparten mis tutores, Sistemas de Almacenamiento y Recuperación de Información, también ha sido muy importante para realizar este trabajo, en ella aprendí a trabajar con grandes cantidades de información, saber procesarlas y trabajar con todos los datos.

En general, para este proyecto he aplicado los conocimientos que he adquirido durante la carrera. Además de los que he mencionado, también he probado funcionalidades de

paralelización que vi en la asignatura de Computación Paralela, he tenido que usar conocimientos de JavaScript para añadir la información a la página web, y por supuesto, todos mis conocimientos de programación.

7.2 Trabajos Futuros

Todos los proyectos tienen que ser actualizados constantemente y añadir nuevas características que se hayan quedado en el tintero o que resuelvan nuevos problemas que surjan con el tiempo. Aquí expondré brevemente los que en mi opinión se podrían abordar en futuras versiones del trabajo.

- Mejorar los resultados de la clasificación, para ello es necesario aumentar la cantidad de sentencias de las que se dispone para el entrenamiento.
- Algunas sentencias hacen referencia a artículos de leyes o referencias a partes anteriores al fallo de la sentencia. En este trabajo no se ha abordado ese problema, pero en el futuro se podría abordar.
- Añadir las ayudas gráficas de cuadros explicativos de conceptos técnicos.

Bibliografía

[1] *¿Qué es la Lectura Fácil (LF)?* Accessed: 2024-06-19. URL: https://www.lecturafacil.net/es/info/1-que-es-la-lectura-facil-lf/.

[2] Elías Esteve Bernal. «Simplificación de sentencias judiciales a Lectura Fácil». Trabajo de Fin de Grado. Universidad Politécnica de Valencia, 2024.

[3] Javier Meliá Sevilla. «Simplificación de textos usando técnicas de procesamiento de lenguaje natural». Trabajo de Fin de Grado. Universidad Politécnica de Valencia, 2021.

[4] *Ley 8/2021, de 2 de junio, por la que se reforma la legislación civil y procesal para el apoyo a las personas con discapacidad en el ejercicio de su capacidad jurídica*. BOE-A-2021-9233. 2021. URL: https://www.boe.es/eli/es/l/2021/06/02/8.

[5] *Fundación Espurna*. Accessed: 2024-06-17. 2024. URL: https://www.espurna.org/.

[6] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-Term Memory». En: *Neural Computation* 9.8 (1997), págs. 1735-1780. DOI: 10.1162/neco.1997.9.8.1735.

[7] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *arXiv preprint arXiv:1810.04805* (2018). URL: https://arxiv.org/abs/1810.04805.

[8] Tom B Brown et al. «Language Models are Few-Shot Learners». En: *arXiv preprint arXiv:2005.14165* (2020). URL: https://arxiv.org/abs/2005.14165.

[9] Ashish Vaswani et al. «Attention is All You Need». En: *Advances in Neural Information Processing Systems* 30 (2017), págs. 5998-6008. URL: https://arxiv.org/abs/1706.03762.

[10] Alexis Conneau et al. «Supervised Learning of Universal Sentence Representations from Natural Language Inference Data». En: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018, págs. 670-680. DOI: 10.18653/v1/d18-2021. URL: https://aclanthology.org/D18-2021.

[11] *Hugging Face*. https://huggingface.co/. Accessed: 2024-06-17.

[12] Recognai. *bert-base-spanish-wwm-cased-xnli*. Accessed: 2024-06-17. 2020. URL: https://huggingface.co/Recognai/bert-base-spanish-wwm-cased-xnli.

[13] Tim Pal. *timpal0l/mdeberta-v3-base-squad2*. Accessed: 2024-06-17. 2021. URL: https://huggingface.co/timpal0l/mdeberta-v3-base-squad2.

[14] Hugging Face. *dccuchile/bert-base-spanish-wwm-cased*. Accessed: 2024-06-17. 2024. URL: https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased.

[15] *Global Symbols*. Accessed: 2024-06-22. 2024. URL: https://globalsymbols.com/?locale=es.

APÉNDICE A
ANEXOS

Anexo I: OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**		x		
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**			x	
ODS 9. **Industria, innovación e infraestructuras.**			x	
ODS 10. **Reducción de las desigualdades.**	x			
ODS 11. **Ciudades y comunidades sostenibles.**			x	
ODS 12. **Producción y consumo responsables.**				x
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**		x		
ODS 17. **Alianzas para lograr objetivos.**		x		

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

- **Reducción de las desigualdades,** el proyecto desarrollado tiene el objetivo principal de ayudar a todas las personas con dificultades en la comprensión lectora mediante un sistema que sintetiza las ideas del fallo de una sentencia judicial en la que estas personas son afectados y necesitan entenderlas.
- **Alianzas para lograr objetivos,** el trabajo desarrollado se lleva a cabo junto a la colaboración de la ONG Espurna, quien se encarga de ayudar a la inserción sociolaboral de las personas con discapacidad intelectual. Esta alianza de la fundación Espurna junto con el VRAIN (*Valencian Research Institute of Artificial Inteligence*) ha hecho posible el desarrollo de la aplicación que se presenta en este trabajo.
- **Paz, justicia e instituciones sólidas,** el trabajo está estrechamente relacionado con la justicia porque permite que los textos jurídicos sean simplificados y tengan mayor alcance. De esta forma, las sentencias, redactadas con lenguaje formal y muy difícil de entender y con oraciones complejas, se resumen en las ideas principales del texto con un lenguaje sencillo que cualquiera puede entender.
- **Salud y bienestar,** otro de los objetivos de la aplicación es mejorar la vida de las personas, concretamente de aquellas con diversidad intelectual.