Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

Lectura Fácil: Grados de Adaptación

Autor: Alonso Martín Arévalo
Tutora: María del Carmen Suárez de Figueroa Baonza
Cotutor: Isam Diab Lozano

Madrid, Enero 2026

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Matemáticas e Informática

Título: Lectura Fácil: Grados de Adaptación

Enero 2026

Autor: Alonso Martín Arévalo
Tutor: María del Carmen Suárez de Figueroa Baonza
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid
Cotutor: Isam Diab Lozano
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

La lectura desempeña un papel fundamental en el desarrollo humano, ya que nos permite acceder a la información, participar activamente en la sociedad y llevar una vida autónoma. No obstante, por diversos motivos, no todas las personas poseen las mismas capacidades de comprensión lectora.

Para responder a esta necesidad, surge la metodología de Lectura Fácil, cuyo propósito es adaptar el lenguaje, la estructura y el contenido de los textos con el fin de facilitar su comprensión a un público más amplio.

El objetivo de este trabajo es diseñar una escala para clasificar textos según su grado de adaptación a la Lectura Fácil, así como una aplicación web que evalúe el grado de adaptación de textos en español mediante un controlador de lógica difusa.

Abstract

Reading plays a fundamental role in human development, as it allows us to access information, actively participate in society and lead an autonomous life. However, for various reasons, not all people have the same reading comprehension abilities.

To respond to this need, the Easy-to-Read methodology arises, whose purpose is to adapt the language, structure and content of the texts in order to facilitate their understanding by a wider audience.

The objective of this work is to design a scale to classify texts according to their degree of adaptation to Easy-to-Read, as well as a web application that evaluates the degree of adaptation of texts in Spanish using a fuzzy logic controller.

Tabla de contenidos

Capítulo 1

Introducción

La lectura desempeña un papel fundamental en el desarrollo del ser humano. Leer nos permite extraer información con la que comprender el mundo que nos rodea, además de reflexionar acerca de nosotros mismos. En definitiva, la lectura y la comprensión lectora son habilidades esenciales que nos permiten llevar una vida autónoma y crecer como seres humanos.

Sin embargo, un notable porcentaje de la población tiene dificultades para comprender textos escritos. De acuerdo con el informe más reciente de la Organización para la Cooperación y el desarrollo Económico (OCDE) [1], *Education at a Glance 2025*, en España un $32\,\%$ de las personas de entre 25 y 64 años tiene un nivel igual o inferior al 1 de un total de 5, es decir, comprenden textos muy breves con información limitada.

Entre las personas con dificultades de comprensión lectora se incluyen aquellas con algún tipo de discapacidad o trastornos de aprendizaje, personas mayores poco alfabetizadas, jóvenes con problemas de escolarización, o inmigrantes que están aprendiendo el idioma, entre otros.

Ante las distintas necesidades lectoras, surge como respuesta la Metodología de Lectura Fácil. En España, esta metodología se rige por la Norma UNE153101:2018 EX [2], que establece las pautas y recomendaciones para la adaptación y elaboración de documentos. Los textos y materiales en Lectura Fácil se adaptan permitiendo una comprensión más simple. Esta adaptación no solo tiene en cuenta el lenguaje, sino que también adapta la maquetación y el contenido, incluidas las imágenes.

Actualmente, se han definido diferentes niveles de adaptación. Según la Federación Internacional de Asociaciones de Bibliotecarios y Bibliotecas (IFLA) se tienen los siguientes niveles:

- Nivel 1: muchas ilustraciones y poco texto, con una complejidad sintáctica y lingüística baja.
- Nivel 2: vocabulario y expresiones de la vida cotidiana, acciones fáciles de seguir y algunas ilustraciones.

- Nivel 3: texto más largo, con palabras poco usuales y sentido figurado, con saltos espacio-temporales y muy pocas ilustraciones.

Otras clasificaciones también han sido desarrolladas en función del número y la frecuencia de las palabras empleadas, o, como en el caso del Marco Común Europeo de Referencia para las Lenguas (MCER) y el Plan Curricular del Instituto Cervantes (PCIC), según el grado de complejidad gramatical [3].

El objetivo de este trabajo, es proponer una escala para la clasificación de textos de acuerdo a su grado de adaptación a la Lectura Fácil, así como desarrollar una aplicación web capaz de obtener el grado de adaptación de textos en español.

Capítulo 2

Estado de la cuestión

2.1. Metodología de Lectura Fácil

La Lectura Fácil, como hemos mencionado en la introducción, busca presentar la información de manera resumida y esquemática, creando así textos accesibles y de fácil interpretación para cualquier persona. El objetivo es que el mayor número de personas pueda acceder a la información para llevar una vida autónoma y crecer como seres humanos. El acceso a la información y contenido cultural es un derecho de todas las personas, y así queda reflejado en la Convención de las Naciones Unidas sobre los Derechos de las Personas con Discapacidad [4].

La metodología de Lectura Fácil está marcada por una serie de pautas, aproximadamente 100 pautas que abarcan vocabulario y expresiones, frases y oraciones, texto y estilo, imágenes y ortotipografía, [2].

2.2. Trabajos previos

Antes de empezar con el desarrollo del trabajo, se han revisado varios trabajos previos en relación con la Lectura Fácil.

Leonardo Pires Morais [5] muestra cómo automatizar el análisis y adaptación de ciertas pautas de la lectura fácil creando servicios web mediante el uso de sPacy y Procesamiento de Lenguaje Natural. Además, implementa este análisis a servicios web previamente desarrollados. Otro trabajo relacionado con la adaptación a la lectura fácil es el realizado por Dayana Danchova Mladenova [6], centrado en el análisis y adaptación oraciones coordinadas explicativas también mediante el uso de sPacy y ofreciendo una interfaz de usuario simple y accesible. Por último, Raúl Rodríguez Martín [7] muestra de manera más profunda y extensa la creación de un servicio web para la lectura fácil.

Asimismo, como referencia para la búsqueda de pautas e información sobre la historia y la actualidad de la Lectura Fácil, se han empleado tanto el libro de Óscar García Muñoz [3], centrado en el análisis y aplicación de la Lectura Fácil, como el blog de Olga Carreras Montoto [8], que ofrece un contexto social actual

a la Lectura Fácil y aporta referencias sobre la normativa Europea acerca de Lectura Fácil.

Además, como referencia para la creación de la escala de clasificación de textos según las pautas de Lectura Fácil, se han tenido en cuenta el libro escrito por Óscar García Muñoz [3], el blog de Olga Carreras Montoto [8] y el libro escrito por Daniel Manrique Gamo y María del Carmen Suárez de Figueroa Baonza [9].

2.3. Herramientas usadas

En esta sección se detallan las herramientas utilizadas para la realización del proyecto, profundizando brevemente en las características de cada una.

2.3.1. Python

Figura 2.1: Icono de *Python*

Python[10] es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, que se caracteriza por una sintaxis clara y sencilla. Es ampliamente utilizado en el desarrollo de aplicaciones relacionadas con la inteligencia artificial y el procesamiento del lenguaje natural, gracias a su extensa colección de bibliotecas y herramientas. Entre las más destacadas en el ámbito de la inteligencia artificial se encuentran *Natural Language Toolkit* (NLTK), *spaCy* y *Scikit-Learn*, entre otras.

2.3.2. spaCy

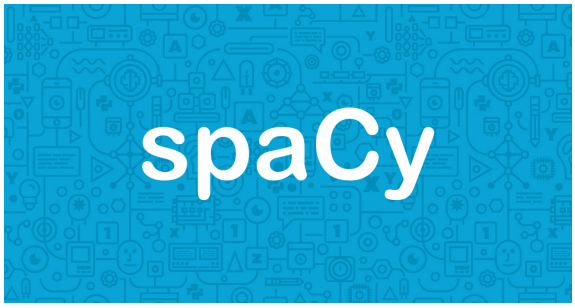


Figura 2.2: Icono de *spaCy*

spaCy[11] es una librería de procesamiento del lenguaje natural (N*atural Language Processing, NLP*) de código abierto. Esta herramienta permite analizar lingüísticamente el texto, segmentarlo en oraciones y palabras, y examinar su estructura gramatical.

Para aprovechar sus funcionalidades, *spaCy* ofrece cuatro modelos preentrenados en español, cada uno con un tamaño y nivel de precisión distintos:

- ***es_core_news_sm***: el modelo más pequeño, usado para tareas ligeras.
- ***es_core_news_md***: el modelo de tamaño medio.
- ***es_core_news_lg***: el modelo de tamaño grande, con mejor precisión que los dos anteriores.
- ***es_core_news_trf***: el modelo transformador basado en arquitecturas de redes neuronales.

Estos modelos incorporan vocabulario, reglas gramaticales y otros recursos lingüísticos específicos del idioma español, permitiendo realizar análisis robustos y eficientes del lenguaje.

2.3.3. scikit-fuzzy

Figura 2.3: Icono de *scikit_fuzzy*

scikit-fuzzy[12] es una colección de algoritmos de lógica difusa diseñados por la comunidad *SciPy* para su uso en *python*.

En este proyecto se ha utilizado *scikit-fuzzy* para implementar el controlador borroso, es decir, la base de reglas y las funciones de pertenencia de cada una de las variables.

2.3.4. PyMuPDF

Figura 2.4: Icono de *PyMuPDF*

PyMuPDF[13] es una librería orientada a la extracción de datos, análisis, conversión y manipulación de archivos *PDF* (.pdf), entre otros, de una manera rápida y ligera.

En este proyecto se ha empleado *PyMuPDF* en la extracción de textos e imágenes de archivos *PDF* para su posterior análisis.

2.3.5. python-pptx y python-docx

Figura 2.5: Icono de Python

python-pptx[14] y *python-docx*[15] son librerías *de Python* utilizadas para la creación, lectura y actualización de archivos *PowerPoint* (.pptx) y *Word* (.docx) respectivamente. También permite la extracción de texto e imágenes, así como la creación de diapositivas o páginas que pueden resultar tediosas de realizar manualmente.

En este proyecto, se han utilizado las librerías para la extracción de textos e imágenes de los archivos *PowerPoint* y *Word* para su posterior evaluación.

2.3.6. os

Figura 2.6: Icono de *os*

os[16] es un módulo en Python que permite usar funcionalidades dependientes del sistema operativo. Entre estas funcionalidades, encontramos la lectura y apertura de archivos, la manipulación de rutas, la creación de archivos temporales o directorios, y el manejo de alto nivel de archivos.

En este proyecto se ha utilizado la librería para la apertura y manipulación de archivos *PDF, Word y PowerPoint*.

2.3.7. Flask

Figura 2.7: Icono de *Flask*

Flask[17] es un *microframework* de código abierto escrito en *Python*, orientado al desarrollo de servicios web y microservicios. Destaca por su ligereza, simplicidad y arquitectura flexible, lo que permite crear aplicaciones web de forma ágil y modular.

Además, *Flask* cuenta con una amplia variedad de extensiones que facilitan la incorporación de funcionalidades adicionales, como la gestión de bases de datos, la autenticación de usuarios o el manejo de formularios, entre otras.

En este proyecto, se ha empleado *Flask* para implementar la aplicación web, gestionar las peticiones de los usuarios y coordinar la comunicación entre la interfaz y los módulos desarrollados en *Python*.

2.3.8. VSCode

Figura 2.8: Icono de *Visual Studio Code*

Visual Studio Code (VSCode)[18] es un editor de código fuente gratuito y de código abierto desarrollado por *Microsoft* para *Windows*, *Linux*, *macOS* y la *Web*. Se ha consolidado como una herramienta esencial para desarrolladores gracias a su amplia gama de características, entre las que destacan una interfaz intuitiva y personalizable, soporte integrado para múltiples lenguajes de programación y un extenso sistema de extensiones que permite ampliar sus funcionalidades.

En este proyecto, se ha empleado *Visual Studio Code* para el desarrollo tanto de la página web como de la aplicación implementada en lenguaje *Python*.

Capítulo 3

Desarrollo

En esta sección de la memoria se describe el proceso de desarrollo del trabajo. La sección 3.1 muestra, de manera general, la estructura de la escala inicial, las características del texto que se tomaron en cuenta para crear la escala y las reglas utilizadas. La sección 3.2 muestra la estructura y la configuración inicial del proyecto. Por último, en la sección 3.3 se muestra la implementación técnica de cada uno de los componentes del motor de inferencia borroso y de la aplicación web.

3.1. Escala inicial

La primera parte del trabajo es la creación de una escala de clasificación basada en los niveles de Lectura Fácil y añadiéndole dos de las pautas adicionales[3]:

- **Nivel 1:** Entre 500 y 1200 palabras con un 10 % de las palabras de frecuencia muy baja, baja o notable en el lenguaje habitual. Las oraciones tienen menos de 15 palabras y abundan las ilustraciones. Las frases cumplen preferentemente la estructura Sujeto-Verbo-Objeto. Se utilizan pocas subordinadas.
- **Nivel 2:** Entre 1200 y 2000 palabras. Alrededor del 10 % del lenguaje puede ser de frecuencia moderada y baja. Las oraciones tienen entre 15 y 20 palabras. Hay algunas ilustraciones. Se limita la complejidad de las oraciones subordinadas.
- **Nivel 3:** Entre 2000 y 3000 palabras con un 10 % de las palabras de frecuencia baja. Las oraciones tienen más de 20 palabras y hay pocas ilustraciones. Se permiten construcciones sintácticas más complejas.
- **Pauta 1:** Se debe evitar el uso de adverbios terminados en *-mente*.
- **Pauta 2:** Se debe evitar el uso de superlativos terminados en *-ísimo*.

Como base para la clasificación de los textos se tomarán las siguientes características:

- Longitud del texto.

- Longitud de la oración.
- Porcentaje de palabras de frecuencia alta o muy alta.
- Número de subordinadas o dependencias.
- Relación entre el número de imágenes y la longitud del texto.
- Relación entre el número de adverbios terminados en *-mente* y el número total de adverbios.
- Relación entre el número de superlativos terminados en *-ísimo* y el número total de adjetivos

El controlador difuso actúa sobre la variable Grado de Pertenencia del texto (*GP*). Esta variable viene determinada por la Longitud de la Oración (*LO*), la Longitud del Texto (*LT*), la Complejidad de las Palabras (CP), el Número de Subordinadas (*NS*) en el texto, la Relación entre Texto e Imágenes (*RTI*), la Relación entre Adverbios y adverbios acabados en *-mente (RAV)* y la Relación entre Adjetivos y superlativos acabados en *-ísimo (RA_isimo)*. Es decir, las siguientes variables:

- *GP*: Grado de Adaptación del texto de acuerdo a los niveles de Lectura Fácil definidos. Sus posibles valores son: Nivel 1 (*N1*), Nivel 2 (*N2*) y Nivel 3 (*N3*). Su intervalo de valores es [0,100] %.
- LO: Longitud de la Oración en función al número medio de palabras de las oraciones del texto . Los valores posibles son: Corta (*C*), Media (*M*) y Larga (*L*). Su intervalo de valores es (0,∞) palabras/oración.
- *LT*: Longitud del Texto en función del número de palabras del texto. Sus posibles valores son: Corto (*CO*), Medio (*ME*) y Largo (*LA*). Su intervalo de valores es (0, ∞) palabras/texto.
- *CP*: Complejidad de las Palabras, que indica el porcentaje de palabras con frecuencia alta o muy alta que aparecen en el texto. Los valores que toma son: Alto (*A*), Moderado (*MO*) y Bajo (*B*). Su intervalo de valores es [0,100]
- *NS*: Número de Subordinadas por cada 100 oraciones en el texto. Los posibles valores son: Pocas (*P*), Algunas (A) y Muchas (M). Su intervalo de valores es (0, ∞) frases/texto.
- *RTI*: Relación entre el número de imágenes y la longitud del texto. Los posibles valores son: Bajo (*BI*), Medio (*MI*) y Alto (*AI*). Su intervalo de valores es $[0, 2500]$ palabras/imagen.
- *RAV*: Relación entre el número de adverbios y los adverbios terminados en *-mente*. Sus posibles valores son Baja (*BADV*), Moderada (*MADV*) y Alta (*AADV*). Su intervalo de valores es $[0, 100]\,\%$
- *RAJ*: Relación entre el número de adjetivos y los superlativos terminados en *-isimo*. Sus posibles valores son Baja (*BADJ*), Moderada (*MADJ*) y Alta (*AADJ*). Su intervalo de valores es $[0, 100]\,\%$

Los valores cualitativos que pueden tomar las variables tienen asociadas las funciones de posibilidad mostradas en las siguientes figuras:

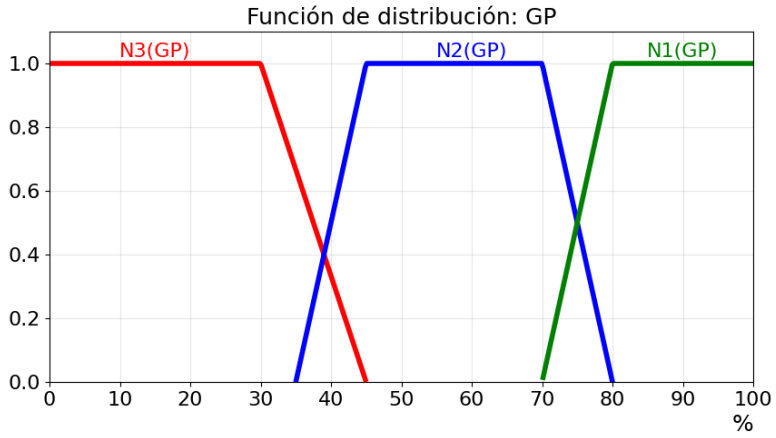


Figura 3.1: Función de distribución *GP*

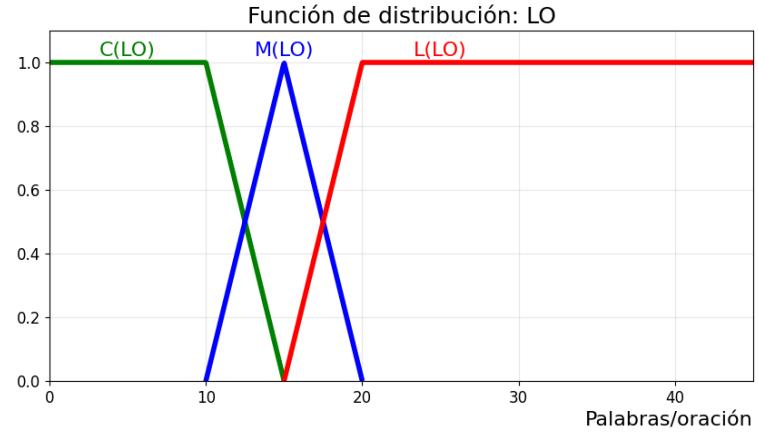


Figura 3.2: Función de distribución *LO*

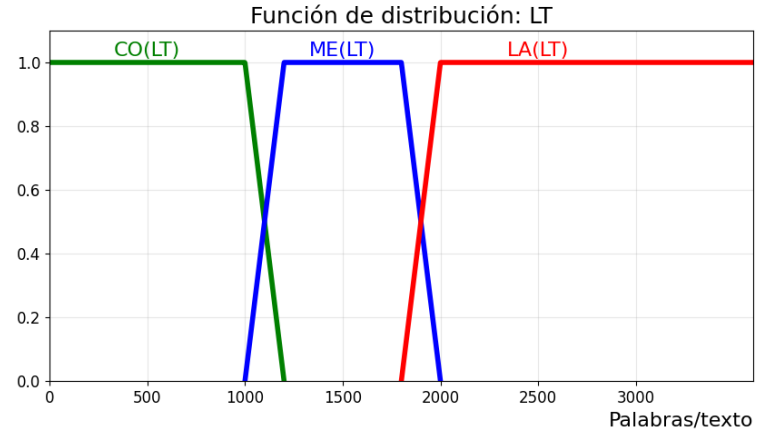


Figura 3.3: Función de distribución *LT*

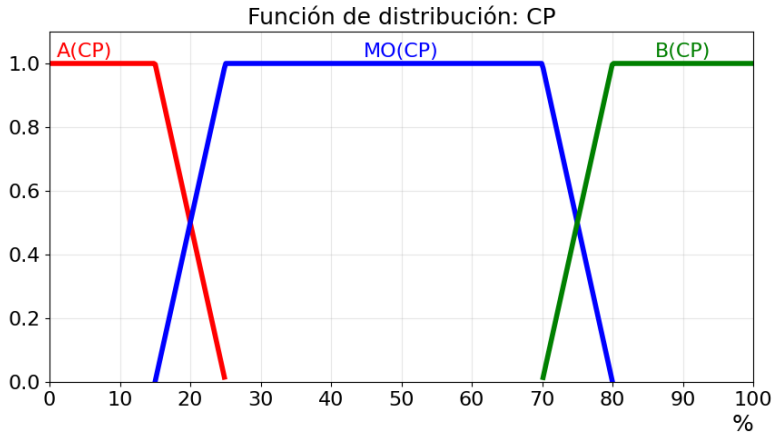


Figura 3.4: Función de distribución *CP*

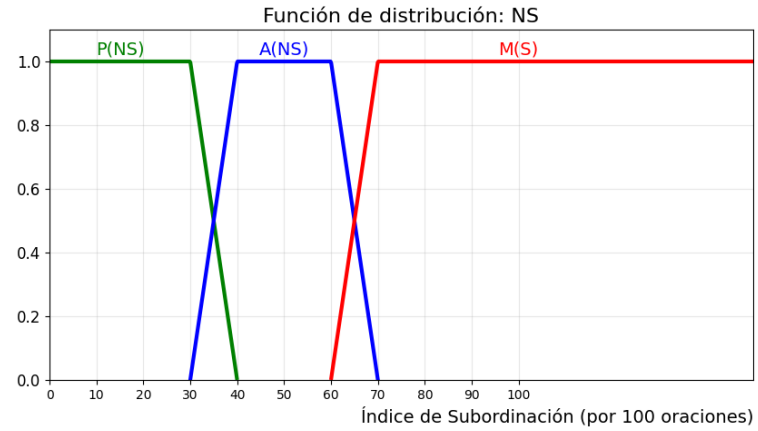


Figura 3.5: Función de distribución *NS*

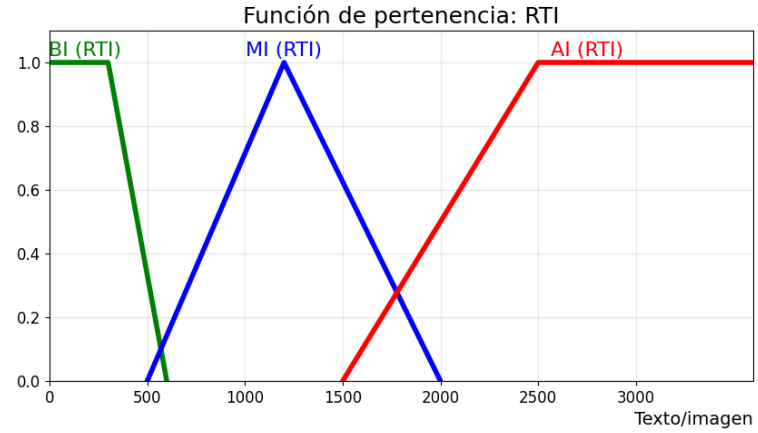


Figura 3.6: Función de distribución RTI

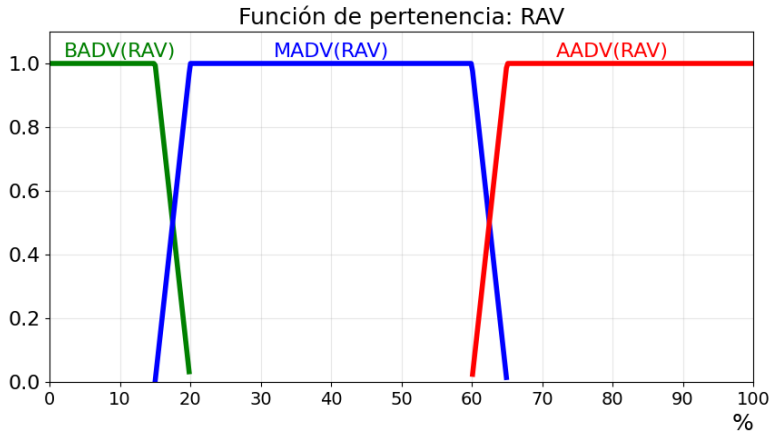


Figura 3.7: Función de distribución RAV

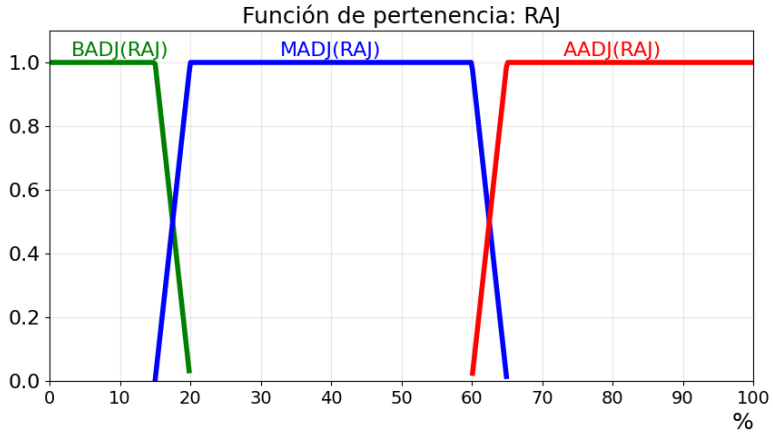


Figura 3.8: Función de distribución RAJ

Además, se tiene la siguiente base de reglas:

Regla	**LO**	**LT**	**CP**	**NS**	**RTI**	**ADV**	**ADJ**	→	**GP**
R1	C	CO	B	P	BI	BADV	BADJ	→	N1
R2	M	CO	B					→	N1
R3	M	CO	MO					→	N1
R4	C	CO	MO					→	N1
R5	C	ME	B	P				→	N1
R6	C	LA	B		AI			→	N1
R7			B	A	BI			→	N1
R8		ME	B		BI			→	N1
R9	M		B		BI			→	N1
R10	M		B			BADV	BADJ	→	N1
R11	C		B	P	BI			→	N1

R12	M	ME	MO	A	MI	MADV	MADJ	→	N2
R13		LA			BI			→	N2
R14	C			M				→	N2
R15	L		B					→	N2
R16		ME		M				→	N2
R17	C	LA	B					→	N2
R18	C	CO				AADV		→	N2
R19	C	CO					AADJ	→	N2
R20	C	CO				MAADV		→	N2
R21	C	CO					MAADJ	→	N2
R22		CO	MO	M	MI			→	N2
R23		CO	MO		AI			→	N2
R24		LA	B	A				→	N2
R25	M		MO	A				→	N2
R26	L	LA	A	M	AI	AADV	AADJ	→	N3
R27						AADV	AADJ	→	N3
R28	L			M				→	N3
R29	L	ME	A					→	N3
R30	M	LA						→	N3
R31	L	LA						→	N3
R32		LA	MO					→	N3
R33		LA	A					→	N3
R34	M	ME				AADV		→	N3
R35	M	ME					AADJ	→	N3
R36		LA				AADV		→	N3
R37		LA					AADJ	→	N3
R38		LA				MAADV		→	N3
R39		LA					MAADJ	→	N3
R40		LA	MO	M	MI			→	N3
R41		ME		M	MI			→	N3
R42		LA		A	AI			→	N3

Cuadro 3.1: Reglas de control difuso

3.2. Estructura del proyecto

El programa se estructura de la siguiente manera:

- **analizadores**
  En la carpeta ***Analizadores*** se encuentran los módulos encargados de analizar el texto, extraer las imágenes y calcular los valores asociados a cada variable. Los módulos son:
  - **complejidad_palabras.py**- clasifica las palabras según su complejidad y calcula el porcentaje de palabras con complejidad muy fácil o fácil.
  - **extraer_texto_imagenes.py**- extrae el texto y el número de imágenes del archivo seleccionado para analizar.

- **longitudes.py**- analiza el texto calculando la longitud total del texto y la longitud media de las oraciones.

- **numero_subordinadas.py**- busca en el texto dependencias que indiquen subordinación y devuelve el número de subordinadas por cada 100 oraciones del texto.

- **relacion_adverbios_superlativos.py** - busca en el texto dependencias que indiquen, o bien adjetivos superlativos acabados en *-isimo*, o bien adverbios acabados en *-mente*, y devuelve la relación entre los adjetivos que se encuentran en el texto y los superlativos acabados en *-isimo* y los adverbios que se encuentran en el texto y los adverbios acabados en *-mente*.

- **relacion_imagenes.py** - toma el número total de imágenes y devuelve la relación entre la longitud del texto y el número total de imágenes.

- **logica_difusa**

  - **evaluador_difuso.py** - contiene la definición de las funciones de pertenencia y la base de reglas. Es el encargado de iniciar el motor de inferencia borroso y evaluar cada variable.

- **modelos**

  - **resultado.py** - este módulo guarda los resultados con el formato del evaluador. Posteriormente, la aplicación web llamará a este módulo para mostrar los resultados.

- **analizador_total.py**
  Este módulo funciona como un módulo principal. Es el encargado de crear los umbrales que usaremos para analizar la complejidad de las palabras, hacer las llamadas necesarias para extraer el texto y las imágenes, *tokenizar* el texto del archivo y llamar al resto de los analizadores.

- **app.py**
  Este módulo es el punto de acceso a la ruta de la aplicación creada con *Flask*. Este recibe el archivo, realiza la llamada al módulo ***analizador_total*** mencionado anteriormente y muestra los resultados. Adicionalmente, junto con el módulo ***app.py***, está la carpeta ***templates*** en la que se encuentra el archivo *HTML* necesario para la creación de la aplicación web.

- **config.py**
  Este módulo contiene las *Url's*, direcciones de archivos y dependencias sintácticas usadas en el resto de módulos.

Para entenderlo de una manera más visual, aquí se muestra una imagen de la estructura:

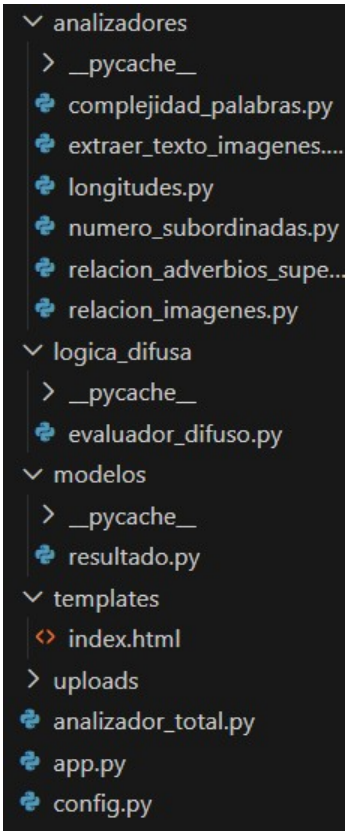


Figura 3.9: Estructura del programa

3.3. Implementación del motor de inferencia borroso

En la siguiente sección, desarrollaré el proceso de implementación del motor de inferencia borroso y las decisiones técnicas detrás de cada variable. Primero, mostraré el proceso de extracción del texto de un archivo. Después, abordaré el análisis de cada una de las variables que influyen en el controlador difuso. Por último, explicaré las reglas seleccionadas.

3.3.1. Extracción de texto e imágenes

El primer paso para poder analizar un archivo es extraer el texto y el número de imágenes. Para ello, se creó el módulo ***extraer_texto_imagenes.py***. Este recibe la ruta absoluta donde se encuentra el archivo. Tras recibir la ruta absoluta, primero decide qué tipo de archivo es: *Word*, *PDF* o *PowerPoint*.

Listing 3.1: Decisión del tipo de archivo

```
def procesar_archivo(self, ruta_archivo):
    # Obtener la extensión
        _, extension = os.path.splitext(ruta_archivo)
        extension = extension.lower()
```

```
        if extension == '.pdf':
            return self.extraer_texto_imagenes_pdf(ruta_archivo)
        elif extension in '.docx':
            return self.extraer_texto_imagenes_docx(ruta_archivo)
        elif extension in '.pptx':
            return self.extraer_texto_imagenes_pptx(ruta_archivo)
```

Después, se recorre el contenido del archivo y se extrae el texto junto con el número de imágenes. De manera que, por ejemplo, para un archivo *PDF*, se implementó la siguiente función:

Listing 3.2: Extracción del texto de un archivo PDF

```
def extraer_texto_imagenes_pdf(self,ruta_archivo):
        try:
            doc = fitz.open(ruta_archivo)
            texto = ""
            total_imagenes = 0
            for pagina in doc:
                texto += pagina.get_text() + "\n"
                imagenes = pagina.get_images()
                cantidad = len(imagenes)
                total_imagenes += cantidad

            doc.close()
            return texto, total_imagenes, None
        except Exception as e:
            return "", 0, f"PDF - Error: {str(e)}"
```

Una vez extraído este texto, mediante el uso de la librería *spaCy*, se procesa el texto. Como mencionábamos en el capítulo 2, esta librería ofrece varios modelos preentrenados. En este caso, usaremos el modelo *es_core_news_lg*, que es el modelo de tamaño grande y con mejor precisión.

Listing 3.3: Procesamiento de un texto

```
nlp = spacy.load("es_core_news_lg")
doc = nlp(texto)
```

Así, el objeto *doc* es el resultado de procesar el texto. Este objeto contiene el texto original, los *tokens* individuales y otras características, como etiquetas gramaticales o relaciones.

3.3.2. Análisis de la complejidad de las palabras

Para analizar la complejidad de las palabras, se ha tomado como referencia el archivo publicado por la Real Academia Española en el Corpus de Referencia del Español Actual con las diez mil formas más frecuentes del español [19]. Este archivo contiene, además de las palabras, su frecuencia absoluta y su frecuencia

normalizada. Para este proyecto, se va a tener en cuenta la frecuencia normalizada, que representa el número de ocurrencias por cada millón de palabras, siendo más representativa que la frecuencia absoluta.

Este tipo de archivos cumple la Ley de Zipf[20]. Esta establece que, dada una lengua cualquiera, la frecuencia de aparición de las distintas palabras sigue una distribución que puede aproximarse a :

$$P_N \sim \frac{1}{n^a}$$

Donde P_n es la frecuencia de una palabra en el orden n y el exponente a es aproximadamente 1.

Para calcular la complejidad de las palabras, se han considerado fáciles o muy fáciles aquellas palabras con mayor frecuencia normalizada, puesto que se puede asumir que son las más comunes. Es por ello que se ha decidido separar las palabras por percentiles y asignarles una categoría a cada una. De manera que las palabras con una frecuencia normalizada mayor que el $95\,\%$ de las palabras reciben la categoría 0; las que tienen una frecuencia normalizada mayor que el $80\,\%$ reciben una categoría 1; las que tienen una frecuencia mayor que el $60\,\%$ reciben una categoría 2; y el resto de las palabras, una categoría 3. Una vez asignadas las categorías, tomaremos solo las palabras de categoría 0 y 1 como palabras muy fáciles o fáciles.

El valor final de la complejidad de las palabras muestra el porcentaje de palabras fáciles que hay en el texto en comparación con su longitud:

$$\text{CP\_val} = \frac{\text{palabras\_faciles}}{\text{palabras\_totales}} \times 100$$

Donde *CP_val* es el valor final de la complejidad de las palabras, *palabras_faciles* el número de palabras fáciles del texto y *palabras_totales* es el número total de palabras del texto.

3.3.3. Análisis de longitudes

Una de las características que deben cumplir los archivos adaptados a Lectura Fácil, de acuerdo con la escala de clasificación, es incluir poco texto. Es decir, el texto debe ser corto. Por ello, una de las variables que se tendrá en cuenta es la longitud del texto. En la implementación, basta con separar el texto en oraciones y, posteriormente, recorrer cada una buscando las palabras de la siguiente manera:

Listing 3.4: Cálculo de la longitud del texto

```
oraciones = list(doc.sents)
palabras = [token.text for token in doc if token.is_alpha
                    or token.like_num]
LT_val = len(palabras)
```

Además de la longitud del texto, otra de las características de los textos es el uso de oraciones cortas. Para usar un valor representativo, tomaremos una media entre la suma de las longitudes de las oraciones y el número total de oraciones:

$$\text{LO\_val} = \frac{\text{longitud\_oraciones}}{\text{oraciones}}$$

Donde *LO_val* es el valor final de la longitud de las oraciones, *longitud_oraciones* es la suma de las longitudes de las oraciones y *oraciones* es el número de oraciones del texto.

3.3.4. Análisis de oraciones subordinadas

Para poder encontrar las subordinadas del texto, haremos uso de las dependencias universales del *framework Universal Dependencies (UD)* [21].

Cuando se procesa el texto, el objeto *doc* contiene cada elemento del texto como un *token*, de manera individual. Estos *tokens*, incluyen el atributo *dep_*, que contiene la dependencia de la palabra en formato *string*.

Dentro de las dependencias universales, aquellas que indican una subordinación son:

- ***advcl*** - *adverbial clause modifier*

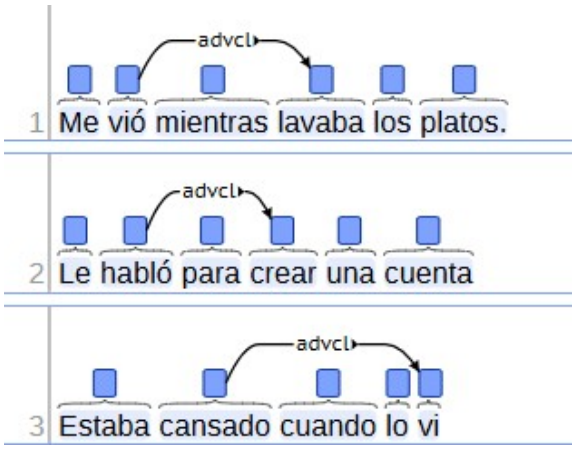


Figura 3.10: Ejemplo de etiqueta *advcl*

- ***csubj*** - *clausal subject*

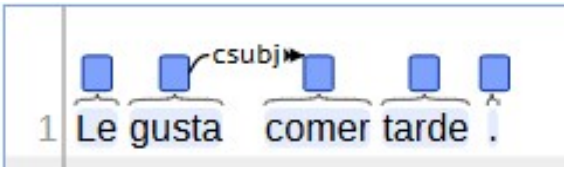


Figura 3.11: Ejemplo de etiqueta *csubj*

- ***ccomp*** - *clasual complement*

Figura 3.12: Ejemplo de etiqueta *ccomp*

- ***xcomp*** - *open clausal complement*

Figura 3.13: Ejemplo de etiqueta *xcomp*

- ***acl*** - *clausal modifier of noun*. Esta última solo se tiene en cuenta si aparece junto con un verbo conjugado y un pronombre relativo.

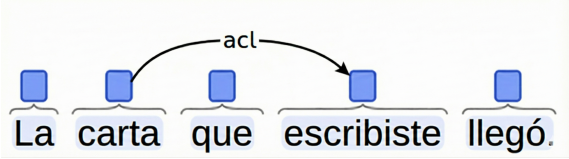


Figura 3.14: Ejemplo de etiqueta *acl*

Para obtener el número de oraciones subordinadas por cada 100 oraciones se realiza la siguiente operación:

$$\text{NS\_val} = \frac{\text{num\_subordinadas}}{\text{oraciones}}$$

Donde *NS_val* es el valor final del número de subordinadas, *num_subordinadas* es el total de subordinadas que contiene el texto y *oraciones* es el número de oraciones del texto.

3.3.5. Análisis de adverbios y adjetivos

Las dos pautas implementadas junto con los tres niveles de adaptación de Lectura Fácil para crear la escala de clasificación son:

1. Evitar el uso de adverbios que terminen en *-mente*.
2. Evitar el uso de adjetivos superlativos que terminen en *-ísimo*.

Para ello, se han utilizado los servicios implementados por el *Ontology Engineering Group* (OEG) de la Universidad Politécnica de Madrid [22]. Estos servicios permiten identificar todos los adverbios terminados en *-mente* y los adjetivos terminados en *-ísimo*. La respuesta que ofrecen los servicios es un diccionario con la siguiente estructura:

Listing 3.5: Cálculo de la longitud del texto

```
    # Frase: "Me encontrarás por aquí frecuentemente"
    {
  'guidelines': [
     {
       'guideline': 'guideline7Vocab',
       'startIndex': 24,
       'endIndex': 38,
       'problematicClause': 'frecuentemente',
       'explanation': 'Evitar␣el␣uso␣de␣adverbios␣terminados␣en␣
          -mente.'
     }
  ]
}
```

Donde *guideline* indica el código de la pauta que no se cumple, *startIndex* y *endIndex* indican los caractéres donde empieza y acaba la palabra, respectivamente, dentro del texto, *problematicClause* indica la palabra que no cumple con la pauta de Lectura Fácil y *explanation* indica cual es la pauta que no se cumple.

Además de identificar los adverbios terminados en *-mente* y los adjetivos terminados en *-ísimo*, vamos se han tenido en cuenta el número total de adverbios y adjetivos del texto. Como mencionábamos en el apartado 3.3.4, el objeto *doc* contiene cada elemento del texto como un *token*. Uno de los atributos de estos *tokens* es *pos_*, que contiene la categoría gramatical de la palabra en formato *string*.

Para tomar un valor representativo, se ha utilizado un porcentaje que indica el número de adverbios terminados en *-mente* respecto al número total de adverbios, y otro que indica el número de adjetivos terminados en *-ísimo* respecto al número total de adjetivos. Es decir:

$$\text{RAV} = \frac{\text{adv\_mente}}{\text{total\_adverbios}} \times 100$$

$$\text{RAJ} = \frac{\text{adj\_isimo}}{\text{total\_adjetivos}} \times 100$$

Donde *RAV* es el porcentaje de adverbios terminados en *-mente*, *adv_mente* es el número total de adverbios que terminan en *-mente*, y *total_adverbios* es el número total de adverbios en el texto. De la misma manera, *RAJ* es el porcentaje de adjetivos terminados en *-isismo*, *adj_isimo* es el número total de adjetivos que terminan en*-isimo* y *total_adjetivos* es el número total de adjetivos en el texto.

3.3.6. Análisis de imágenes

El uso de imágenes e ilustraciones es clave en la adaptación de archivos a la Lectura Fácil. Es por ello, que se ha incluido como una de las variables que actúan sobre el motor de inferencia borroso.

Tras procesar el archivo en el módulo *extraer_texto_imagenes.py*, y contar el número de imágenes que contiene, comparamos el número de imágenes con la

longitud del texto. Es decir:

$$\text{RTI\_val} = \frac{\text{LT\_val}}{\text{num\_imagenes}}$$

Donde *RTI_val* indica la relación entre texto e imágenes del archivo, *LT_val* es la longitud del texto y *num_imagenes* es el número de imágenes que contiene el archivo. Si el archivo no tuviese imágenes, se le asignaría a *RTI_val* el máximo valor posible, lo que indicaría que no incluye imágenes.

3.3.7. Evaluador difuso

Implementación del evaluador difuso

En el apartado 3.1, se definió la escala inicial, las variables a tener en cuenta para nuestro controlador difuso y las reglas que las relacionan entre sí. Ahora, trasladamos las funciones de posibilidad, los dominios de las variables y las reglas que las relacionan a nuestro módulo *evaluador_difuso*.

Las reglas borrosas que relacionan las variables entre sí siguen la siguiente sintaxis:

$$\text{antecedente} \rightarrow \text{consecuente}$$

El antecedente está formado por un conjunto de cláusulas de la forma *variable = valor* o *variable es valor*, donde *valor* es cualitativo al tratarse de un controlador borroso. Las cláusulas del antecedente se relacionan entre sí mediante los operadores lógicos de conjunción ($\wedge$) o disyunción ($\vee$). Además, cada una de las cláusulas puede estar precedida del operador lógico de negación ($\neg$). El consecuente está formado por una única cláusula de la forma *variable = valor* o *variable es valor*. El antecedente y el consecuente se relacionan entre sí mediante el operador lógico de implicación ($\rightarrow$). Por tanto, una regla para nuestro controlador borroso tendrá la siguiente representación:

$$[N]A_1(x_1)\ op_1\ [N]A_2(x_2)\ op_2 \ldots op_{n-1}\ [N]A_n(x_n) \rightarrow B(y)$$

Donde $op_1 \ldots op_{n-1}$ son los operadores borrosos de conjunción o disyunción, *[N]* representa la opción de que la siguiente cláusula sea afectada por la negación, $\rightarrow$ representa el operador lógico de implicación y las etiquetas lingüísticas $A_1 \ldots A_n, B$ están definidas mediante sus correspondientes funciones de distribución de posibilidad.

Es decir, si suponemos la regla:

'Si un texto es corto y sus oraciones son cortas y la mayor parte de sus palabras tienen una frecuencia de uso alta y usa pocas subordinadas y contiene muchas imágenes y no usa pocos adverbios terminados en -mente y usa pocos superlativos terminados en -ísimo entonces el grado de adaptación a la lectura fácil es de nivel 1'

La representación sería:

$$\text{corto(LT)} \wedge \text{corta(LO)} \wedge \text{alto(CP)} \wedge \text{pocas(NS)} \wedge \text{bajo(RTI)} \\ \wedge \neg\text{bajos(RAV)} \wedge \text{baja(RAJ)} \rightarrow \text{N1(CP)}$$

Sin embargo, para facilitar el proceso de inferencia, las reglas del controlador borroso están siempre unidas por conjunciones. De manera que las reglas del controlador seguirán esta forma:

$$R_i : A_i^1(x_i) \wedge \cdots \wedge A_i^n(x_n) \rightarrow D_i(v)$$

Donde cada etiqueta lingüística A_i^i, D_i esta definida por su correspondiente función de posibilidad.

Mediante el uso de la librería *scikit-fuzzy* , podemos definir el dominio de valores que puede tomar cada variable, su significado y si se trata de un antecedente o un consecuente de la siguiente manera:

Listing 3.6: Implementación de antecedentes y el consecuente

```
self.LO = ctrl.Antecedent(np.arange(0, 41, 1), 'LO')
self.LT = ctrl.Antecedent(np.arange(0, 3001, 1), 'LT')
self.CP = ctrl.Antecedent(np.arange(0, 101, 1), 'CP')
self.NS = ctrl.Antecedent(np.arange(0, 101, 1), 'NS')
self.RTI = ctrl.Antecedent(np.arange(0, 5001, 1), 'RTI')
self.ADV = ctrl.Antecedent(np.arange(0, 101, 1), 'ADV')
self.ADJ = ctrl.Antecedent(np.arange(0, 101, 1), 'ADJ')

self.GP = ctrl.Consequent(np.arange(0, 101, 1), 'GP')
```

Por otro lado, la misma librería permite crear las funciones de distribución de posibilidad para cada variable. Por ejemplo, para la variable *LO* correspondiente a la longitud de la oración:

Listing 3.7: Implementación de la función de distribución de posibilidad

```
self.LO['corta'] = fuzz.trapmf(self.LO.universe, [0, 0, 10, 15])
self.LO['media'] = fuzz.trimf(self.LO.universe, [10, 15, 20])
self.LO['larga'] = fuzz.trapmf(self.LO.universe, [15, 20, 41, 41])
```

Por último, una vez introducidos los antecedentes, los dominios de cada variable, sus respectivos significados y las funciones de distribución de posibilidad, se introducen las reglas que conforman la base de conocimiento del controlador.

Listing 3.8: Ejemplo de implementación de las reglas

```
rules = []

    rules.append(ctrl.Rule(
        self.LO['corta'] & self.LT['corto'] & self.CP['baja'] &
           self.NS['pocas'] &
        self.RTI['bajo'] & self.ADV['baja'] & self.ADJ['baja'],
        self.GP['nivel1']
    ))

    self.system = ctrl.ControlSystem(rules)
```

Proceso de inferencia

Con la base de conocimiento formada, se completa el motor de inferencia borroso. Ahora, la librería *scikit-fuzzy* realiza de manera automática el proceso de inferencia. A continuación, se describe como sería este proceso.

En la lógica proposicional, se utiliza como mecanismo de inferencia el *modus ponens*. Este expresa que, dada una regla cierta A $\rightarrow$ B, si se cumple A, entonces se deduce que se cumple B.

En lógica borrosa, se generaliza el *modus ponens*, ya que no es necesario que se cumpla el hecho A, sino que es suficiente con que haya una intersección no vacía entre el antecedente de la regla A y el hecho que se cumple A', dado que ambos son conjuntos borrosos. Entonces, tomando ahora los conjuntos borrosos A, B, A' y B', dada la regla A $\rightarrow$ B, si en vez de A se cumple un hecho A' parecido con intersección no vacía con A, no tiene que cumplirse B, pero si un hecho parecido, en este caso B'. Es decir:

Regla borrosa:	$\mu_A(x) \rightarrow \mu_B(y)$
Hecho:	$\mu_{A'}(x)$
Resultado:	$\mu_{B'}(y)$

Para calcular la función de distribución de posibilidad resultado del proceso de inferencia, $\mu_{B'}(y)$, realizado a partir de una base de conocimiento de reglas de la forma $\mu_A(x) \rightarrow \mu_B(y)$, y conociendo que se cumple el hecho A', representado por $\mu_{A'}(x)$, se utiliza el procedimiento denominado Regla Composicional de Inferencia (RCI):

$$\mu_{B'}(y) = Sup_x \left\{ T\left(\mu_{A'}(x), \mu_{A \rightarrow B}(x, y)\right) \right\}$$

Para calcular los resultados, usaremos las siguientes funciones y la operación extensión cilíndrica:

- **Extensión cilíndrica:** En general, dadas dos distribuciones de posibilidad $\mu_p(x)$ y $\mu_q(y)$, donde x e y pueden ser n-dimensionales, no es posible componerlas porque ambas deben estar definidas en función de las mismas variables. Entonces, para una distribución de posibilidad $\mu_p(x_1, x_2, \ldots, x_n)$, se define la operación extensión cilíndrica de $\mu_p(x_1, x_2, \ldots, x_n)$ con y a la distribución de posibilidad $\mu_p(x_1, x_2, \ldots, x_n, y)$ tal que:

  $$\forall y, \quad \mu_p(x_1, x_2, \ldots, x_n, y) = \mu_p(x_1, x_2, \ldots, x_n)$$

  Sean ahora M el conjunto de todas las funciones de distribución de posibilidad expresadas en función de la variable o vector $x \in \mathcal{X}$, siendo $\mathcal{X}$ el universo del discurso.

  Consideremos cuatro funciones de pertenencia $\mu_p(x), \mu_q(x), \mu_r(x), \mu_s(x) \in M$. Para facilitar la lectura de las propiedades axiomáticas, se empleará la siguiente notación simplificada:

  $$\mu_p(x) \equiv \dot{x}, \quad \mu_q(x) \equiv \dot{y}, \quad \mu_r(x) \equiv \dot{z}, \quad \mu_s(x) \equiv \dot{t}$$

- **T-norma:** una función T que modela la intersección (AND), definida como:

$$T : M^2 \to M$$
$$(\dot{x}, \dot{y}) \mapsto T(\dot{x}, \dot{y})$$

Las funciones t-norma deben cumplir las siguientes propiedades $\forall \dot{x}, \dot{y}, \dot{z}, \dot{t} \in M$:

- **Propiedad conmutativa:** $T(\dot{x}, \dot{y}) = T(\dot{y}, \dot{x})$
- **Propiedad asociativa:** $T(\dot{x}, T(\dot{y}, \dot{z})) = T(T(\dot{x}, \dot{y}), \dot{z})$
- **Elemento neutro:** Dado el valor 1 (pertenencia total), se cumple que $T(\dot{x}, 1) = \dot{x}$
- **Monotonía:** Si $\dot{x} \leq \dot{z}$ y $\dot{y} \leq \dot{t} \implies T(\dot{x}, \dot{y}) \leq T(\dot{z}, \dot{t})$

Aunque existen diversas t-normas, en este proyecto se usará la t-norma del mínimo (o de Gödel):

$$T(\dot{x}, \dot{y}) = \text{mín}(\dot{x}, \dot{y})$$

- **S-norma (o T-conorma):** una función S que modela la unión (OR), definida como:

$$S : M^2 \to M$$
$$(\dot{x}, \dot{y}) \mapsto S(\dot{x}, \dot{y})$$

Las funciones s-norma deben cumplir las siguientes propiedades $\forall \dot{x}, \dot{y}, \dot{z}, \dot{t} \in M$:

- **Propiedad conmutativa:** $S(\dot{x}, \dot{y}) = S(\dot{y}, \dot{x})$
- **Propiedad asociativa:** $S(\dot{x}, S(\dot{y}, \dot{z})) = S(S(\dot{x}, \dot{y}), \dot{z})$
- **Elemento neutro:** Dado el valor 0 (pertenencia nula), se cumple que $S(\dot{x}, 0) = \dot{x}$
- **Monotonía:** Si $\dot{x} \leq \dot{z}$ y $\dot{y} \leq \dot{t} \implies S(\dot{x}, \dot{y}) \leq S(\dot{z}, \dot{t})$

Siguiendo la lógica dual, para este proyecto se usará la s-norma del máximo:

$$S(\dot{x}, \dot{y}) = \text{máx}(\dot{x}, \dot{y})$$

- **Negación -** una función N, tal que $\forall \mu_p(x) \in M$, se define la aplicación para todo $x \in \mathcal{X}$ como:

$$N : M \to M$$
$$\mu_p(x) \mapsto \mu_{\neg p}(x)$$

La función de negación, aplicada a funciones de distribución de posibilidad, cumple las mismas propiedades axiomáticas. Para todo $\dot{x}, \dot{y} \in M$, y representando las funciones constantes $\mu_0(x) = 0$ como 0 y $\mu_1(x) = 1$ como 1:

- **Condiciones frontera:** $N(0) = 1; \quad N(1) = 0$
- **Inversión de monotonía:** Si $\forall x \in \mathcal{X},\ \dot{x} \leq \dot{y} \implies N(\dot{x}) \geq N(\dot{y})$

Existen varios ejemplos de funciones de negación; para este proyecto se utilizará la siguiente:

$$N(\dot{x}) = 1 - \dot{x}$$

- **Función de Implicación (J) -** modela la relación causal "Si p entonces q". Se define como una función J:

$$J : M^2 \to M$$
$$(\dot{x}, \dot{y}) \mapsto J(\dot{x}, \dot{y})$$

Aunque existen diversas formas de modelar la implicación (Larsen, Zadeh, Lukasiewicz...), en este proyecto se utilizará la implicación de Mamdani (o implicación del mínimo). Esta interpreta la regla "Si A entonces Bçomo una correlación o intersección, truncando el conjunto de salida:

$$J(\dot{x}, \dot{y}) = \text{mín}(\dot{x}, \dot{y})$$

Al tratarse de un controlador borroso, la regla composicional de inferencia sufre unos cambios:

- La distribución de posibilidad $\mu_{B'}(y)$ se expresa como $D'(v)$.
- x es un vector del mismo tamaño que variables contiene el controlador, es decir, n-dimensional. La distribución de posibilidad de $\mu_{A'}(x)$ se calcula con la t-norma de las funciones de posibilidad que toman las variables de estado en un momento dado: $A^{1'}(x_1), A^{2'}(x_2), \ldots, A^{n'}(x_n)$. El resultado de esta t-norma representa el resultado de la conjunción $A'_1 \wedge A'_2 \wedge \cdots \wedge A'_n$ que son los valores cualitativos que toman las variables de estado en un momento dado.
- $\mu_{A \to B}$ es $R_i : A_1(x_1) \wedge \cdots \wedge A_n(x_n) \to D_i(v)$, para la regla i-ésima del controlador borroso

Quedando la regla composicional de inferencia como:

$$D'_i(v) = Sup_x\{T((A^{1'}(x_1) \wedge \cdots \wedge A^{n'}(x_n), (A^1_i(x_1) \wedge \cdots \wedge A^n_i(x_n) \to D_i(v)))\}$$

Puesto que el controlador borroso utiliza la implicación de Mamdani y la t-norma del mínimo, se puede simplificar la regla composicional de inferencia, ya que las conjunciones e implicaciones se resuelven mediante la función mín(x,y).

Al sustituir las conjunciones y la implicación por la función mín(x,y), queda la siguiente expresión:

$$\begin{aligned} D'_i(v) = Sup_x\{&\text{mín}(\text{mín}(A^{1'}(x_1), \ldots, A^{n'}(x_n)), \\ &\text{mín}(A^1_i(x_1), \ldots, A^n_i(x_n), D_i(v)))\} \end{aligned}$$

Y aplicando la propiedad asociativa, quedaría así:

$$\begin{aligned} D'_i(v) = \text{mín}(\text{mín}(\ldots \text{mín}(&Sup_{x_1}\text{mín}(A^{1'}(x_1), A^1_i(x_1)), \\ &Sup_{x_2}\text{mín}(A^{2'}(x_2), A^2_i(x_2))), \ldots), D_i(v)) \end{aligned}$$

De esta manera, para cada regla calcularíamos su nivel de ajuste de la siguiente manera:

$$Sup_{x_i} = \{\text{mín}(A^{j'}(x_j), A_i^j(x_j)\} = NA_{j,i}$$

Por tanto:

$$D_i'(v) = \text{mín}(\text{mín}(NA_{1,i}, NA_{2,i}, \dots NA_{n,i}), D_i(v))$$

Y si consideramos $\text{NA}_i = \text{mín}(NA_{1,i}, NA_{2,i}, \dots NA_{n,i})$, entonces:

$$D_i'(v) = \text{mín}(NA_i, D_i(v))$$

Donde NA_i se debe interpretar como una función de distribución de posibilidad que devuelve el valor NA_i para todo el dominio.

Tras este proceso, para cada una de las reglas R_i se obtiene una función de distribución de posibilidad del consecuente: $D_i'(v)$. Por tanto, es necesario calcular la función de distribución de posibilidad final $D'(v)$. Para ello, se calcula la s-norma de las funciones de distribución de posibilidad $D_i'(v) \forall i$.

Este proceso de inferencia se realiza automáticamente mediante la librería *scikit-fuzzy* , utilizando las mismas funciones t-norma, s-norma, de negación e implicación mencionadas anteriormente.

Interpretación del resultado

Tras el proceso de inferencia, es necesario hacer un proceso de interpretación o conversión de $D'(v)$ a un número mediante el cálculo del centro de gravedad, o a una etiqueta lingüística mediante el método de la distancia o la Σ_{cuenta}.

La librería *skfuzzy* trabaja con el método del centro de gravedad para devolver un valor nítido.

Para el cálculo del centro de gravedad, se eligen unos puntos de discretización $\{v_1, v_2, \dots, v_n\}$ a intervalos regulares. Después, se calcula el centro de gravedad como:

$$g_{D'} = \frac{\sum_{j=1}^{m} v_j \cdot D'(v_j)}{\sum_{j=1}^{m} D'(v_j)}$$

De esta manera, se obtiene un valor nítido para clasificar los archivos.

3.4. Implementación de la aplicación web

Para facilitar al usuario el uso del controlador difuso, se ha creado una aplicación web. La página ha sido construida con el *microframework Flask*, utilizando componentes *HTML* y *CSS* simples.

El archivo *index.html* contiene la interfaz, que consiste en un cuadro donde introducir el archivo a analizar y otro donde se mostrarán los resultados. Al seleccionar el archivo a analizar y presionar el botón '*Analizar Documento*', la aplicación realiza la llamada al analizador.

Listing 3.9: Código del archivo app.py

```
from flask import Flask, render_template, request
from analizador_total import AnalizadorLecturaFacil

app = Flask(__name__)
analizador = AnalizadorLecturaFacil()

@app.route("/", methods=["GET", "POST"])
def index():
    resultados = None
    total_imagenes = None

    if request.method == "POST":
        archivo = request.files.get("archivo")
        if archivo:
            try:
                resultados, total_imagenes = analizador.
                    analizar_texto(archivo)
            except Exception as e:
                return f"<h2>Error␣al␣procesar␣el␣archivo:</h2><
                    pre>{e}</pre>"

    return render_template(
        "index.html",
        resultados=resultados,
        total_imagenes=total_imagenes
    )

if __name__ == "__main__":
    app.run(debug=True)
```

El analizador devuelve los valores de cada variable que se muestran en el recuadro reservado para los resultados.

Listing 3.10: Código dentro del archivo index.html

```
{% if resultados %}
        <div class="result-item"><b>Longitud media de oraciones
            (LO):</b> <span>{{ resultados.LO | round(2) }}</span>
            </div>
        <div class="result-item"><b>Longitud del texto (LT):</b>
             <span>{{ resultados.LT }}</span></div>
        <div class="result-item"><b>Complejidad de las palabras
            (CP):</b> <span>{{ resultados.CP | round(2) }} %</
```

```
            span></div>
        <div class="result-item"><b>Número de subordinadas (NS):
            </b> <span>{{ resultados.NS }}</span></div>
        <div class="result-item"><b>Relación Texto-Imagen (RTI):
            </b> <span>{{ resultados.RTI | round(2) }}</span></
            div>
        <div class="result-item"><b>Adverbios acabados en -mente
             (ADV):</b> <span>{{ resultados.ADV | round(2) }} %</
            span></div>
        <div class="result-item"><b>Adjetivos acabados en -ísimo
             (ADJ):</b> <span>{{ resultados.ADJ | round(2) }} %</
            span></div>

        <div class="general-score">
        Grado de Pertenencia a la Lectura Fácil:
        <span>{{ resultados.GP | round(2) }} %</span>
        </div>
{% endif %}
```

De manera que la interfaz se vería de la siguiente manera:

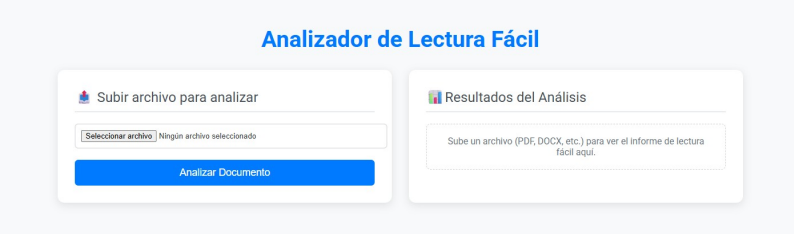


Figura 3.15: Pantalla de inicio

Figura 3.16: Pantalla de resultados

Capítulo 4

Pruebas

La validación del software es una fase fundamental para garantizar tanto su correcto funcionamiento como su utilidad práctica. En este proyecto, se han llevado a cabo diversas pruebas utilizando archivos con diferentes características y formatos. Entre ellos, se han incluido documentos previamente adaptados a Lectura Fácil. Estos textos provienen de fuentes de referencia en la Lectura Fácil, tales como las plataformas web de Plena Inclusión [23] y Planeta Fácil [24], así como la Biblioteca Social Educativa (BASE) de la Sociedad Insular para la Promoción de las Personas con Discapacidad (Sinpromi) [25].

4.1. Pruebas con archivos *PDF*

4.1.1. Prueba 1

En este primer caso, se evalúa un archivo publicado en la plataforma web de Plena Inclusión de la Comunidad de Andalucía [26].

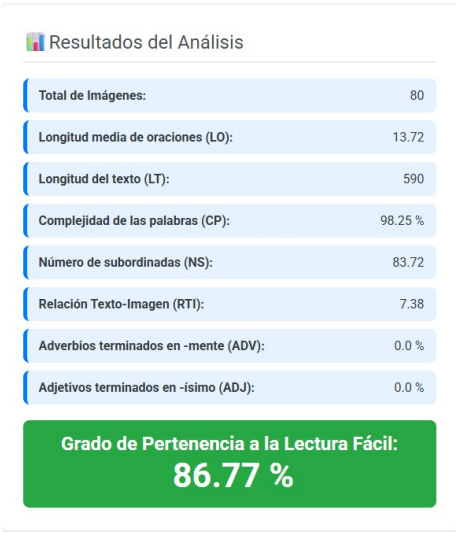


Figura 4.1: Resultados de analizar el archivo Plena Inclusión

Al tratarse de un texto adaptado a la Lectura Fácil, tiene una longitud corta, oraciones cortas, un vocabulario simple y está acompañado de muchas imágenes. Además, cumple con las pautas relacionadas con el uso de adverbios terminados en *-mente* y adjetivos terminados en *-isimo*. Aunque se detectan varias oraciones subordinadas, el resto de las variables corresponde claramente a un texto adaptado. En efecto, el resultado que nos proporciona la aplicación es que el archivo tiene un 86.77 % de adaptación a la Lectura Fácil. Los resultados obtenidos son los siguientes:

4.1.2. Prueba 2

El siguiente archivo es el cuento adaptado *La ultima hoja* de O.Henry, publicado por la Sociedad Insular para la Promoción de las Personas con Discapacidad [27]. Los resultados obtenidos son:

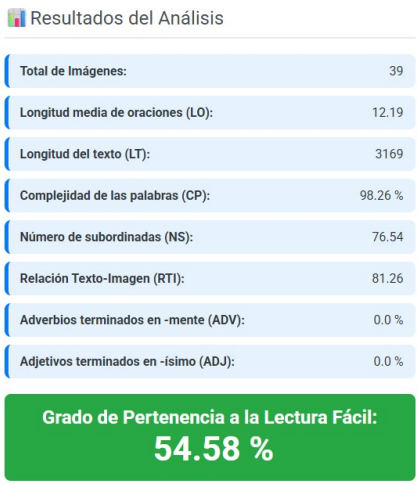


Figura 4.2: Resultados del análisis del cuento *La última hoja*

Si analizamos los resultados, observamos que todas las variables pertenecen a un nivel 1 de pertenencia de adaptación, excepto dos: la longitud del texto y el número de oraciones subordinadas. El analizador, al detectar una longitud del texto muy grande y muchas subordinadas, sitúa el archivo en un nivel 2 de pertenencia con un 54.58 % de valoración.

4.1.3. Prueba 3

El último archivo en formato *PDF* es la Constitución Española publicada en el Boletín Oficial del Estado [28]. Los resultados obtenidos son:

Al tratarse de un texto legislativo, su longitud y el uso de imágenes e ilustraciones lo convierten en un texto complejo de leer. Aun así, su porcentaje de adaptación no es excesivamente bajo, ya que se utilizan oraciones cortas con un vocabulario sencillo.

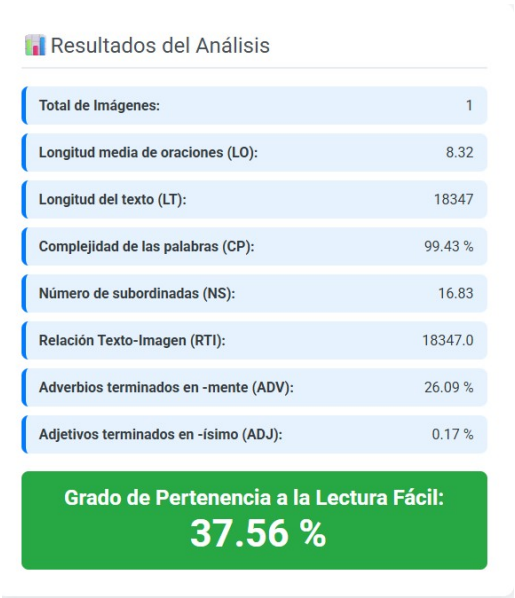


Figura 4.3: Resultados del análisis de la Constitución Española

4.2. Pruebas con archivos *Word*

4.2.1. Prueba 1

El siguiente archivo que se analiza es la guía *Pensar en igualdad: guía en Lectura Fácil* publicada en la Biblioteca Social Educativa (BASE) [29]. Este texto está adaptado según las pautas de Lectura Fácil. Los resultados son los siguientes:

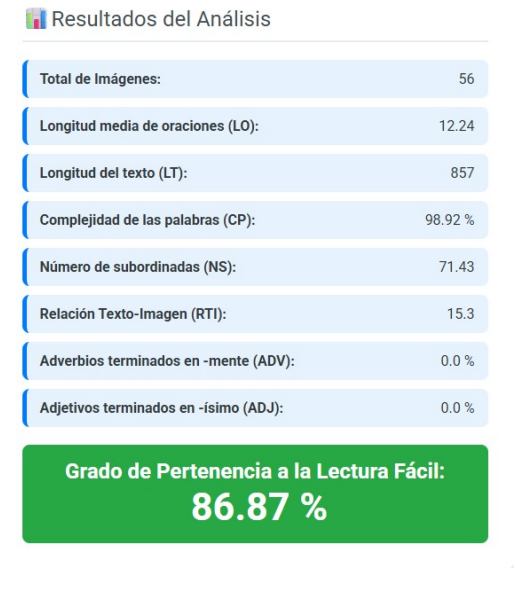


Figura 4.4: Resultados del análisis de *Pensar en igualdad: guía en Lectura Fácil*

Al igual que en el primer archivo *PDF*, encontramos que todas las variables

indican un nivel 1 de adaptación, excepto el número de oraciones subordinadas, que es elevado. Es por ello que se le asigna un valor de 86.87 %, y por lo tanto, un nivel 1 de adaptación.

4.2.2. Prueba 2

Otro archivo en formato *Word* analizado ha sido el cuento *Hoichi: cuento japonés de fantasmas en Lectura Fácil* de Lafcadio Hearn. Este también se trata de un archivo adaptado. Estos son los resultados:

Figura 4.5: Resultado del análisis del cuento *Hoichi: cuento en japonés de fantasmas*

Este cuento, a pesar de estar adaptado, es un texto largo y contiene un alto número de oraciones subordinadas. Sin embargo, el uso de vocabulario simple y numerosas imágenes e ilustraciones contrarresta estas variables, y lo sitúa en un grado de adaptación nivel 2 con un valor de 68.89 %.

4.2.3. Prueba 3

Como último archivo en formato *Word* se analiza la edición del ensayo *Ideas y Creencias* de Ortega y Gasset, publicado en la Biblioteca Virtual OMEGALFA [30].

El ensayo es un texto largo que no contiene ningún tipo de imagen, tiene numerosas subordinadas y oraciones largas. Es por eso que se le califica con un valor del 39.72 % de adaptación. El resultado se encuentra entre el nivel 1 y 2 de adaptación, ya que usa un vocabulario simple.

Figura 4.6: Resultado del análisis del ensayo *Ideas y Creencias*

4.3. Pruebas con archivos *PowerPoint*

4.3.1. Prueba 1

El primer archivo a analizar es una presentación de la Universidad Autónoma del Estado de Hidalgo sobre la ciencia de la historia [31]. Este es el análisis obtenido:

Figura 4.7: Resultado del análisis de la presentación de la Universidad Autónoma del Estado de Hidalgo

La presentación contiene numerosas imágenes y un vocabulario sencillo, es por ello que tiene un porcentaje de grado de pertenencia a la lectura fácil del 79.85 %.

4.3.2. Prueba 2

El segundo archivo es una presentación de la Facultad de Historia de la Universidad Veracruzana sobre Polibio de Megápolis [32].

Figura 4.8: Resultado del análisis de la presentación de la Universidad Veracruzana

En este caso, encontramos un texto largo con un vocabulario simple pero con frases complejas. Esto justifica un grado de pertenencia bajo que lo sitúa como un archivo de nivel 3.

Capítulo 5

Conclusiones y trabajos futuros

Los resultados de las pruebas realizadas con el controlador difuso implementado muestran que, teniendo en cuenta las variables y las reglas aplicadas, los resultados son correctos. Sin embargo, en una visión general, la clasificación y el valor asignados a los textos no siempre son fieles a la realidad.

Si bien los casos extremos, como los textos cortos adaptados o los textos legislativos, se analizan y clasifican con un grado de adaptación de nivel 1 (fáciles) o nivel 3 (complejos) respectivamente, no ocurre lo mismo con los textos con características difusas. Por ejemplo, los cuentos adaptados según las pautas de la Lectura Fácil reciben una valoración de nivel 2 (intermedio), muy cercana al Nivel 3, a pesar de estar adaptados para recibir una clasificación cercana al nivel 1. Esto se debe a que la longitud del texto completo y el número de oraciones subordinadas suelen ser elevados en las historias, lo que penaliza la valoración de estos textos a pesar de tener un vocabulario simple y numerosas imágenes.

Si se restringe el análisis de textos a las variables definidas, los resultados son lógicos, sin embargo, hay varias mejoras que se podrían llevar a cabo en futuras líneas de trabajo.

Por un lado, la modificación de la escala. Además de las variables, pautas y restricciones que incluye la escala actual, se puede modificar la escala añadiendo nuevas restricciones o tomando nuevas pautas, como las relacionadas con la escritura de números, horas, años, extranjerismos, acrónimos, etc.

Por otro lado, las variables y restricciones de la escala pueden sufrir modificaciones para calcular valores más representativos. También se pueden añadir, eliminar o reescribir las reglas actuales, o incluir otras nuevas para hacer más claro el análisis de textos con características que varían entre los diferentes niveles de adaptación.

Además, se podría extender la funcionalidad de la aplicación para procesar nuevos formatos de archivo y analizar páginas *web*. Esto último haría que la aplicación fuera más útil en entornos digitales.

En cuanto a los resultados obtenidos por la aplicación, sería conveniente imple-

mentar los métodos de la distancia o de la Σ_{cuenta} para calcular y mostrar las etiquetas lingüísticas correspondientes a los archivos analizados.

Capítulo 6

Análisis de impacto

Desde el inicio del proyecto, se ha priorizado la necesidad de adaptar la información de acuerdo con las pautas de Lectura Fácil. A través de estas adaptaciones, se garantiza el acceso a la cultura y al desarrollo de una vida autónoma a cualquier persona.

A partir del año 2026, la implementación de la Lectura Fácil en entornos web y de medidas de accesibilidad digital será obligatoria por ley para el sector público y parte del sector privado, de acuerdo con lo establecido en el Real Decreto 1112/2018 [33] sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público, y en cumplimiento con la Directiva (UE) 2019/882 [34] del Parlamento Europeo y del Consejo sobre los requisitos de accesibilidad de los productos y servicios. Este trabajo busca avanzar en la implementación de dichas medidas orientadas al bienestar social.

En relación con los Objetivos de Desarrollo Sostenible incluidos en la Agenda 2030 aprobada por la Organización de las Naciones Unidas (ONU) [35], este trabajo se enmarca tanto en el cuarto objetivo, que promueve una educación de calidad e inclusiva, como en el décimo, el cual busca acabar con las desigualdades. El proyecto ofrece una nueva herramienta para facilitar el derecho universal a la educación, independientemente de la capacidad cognitiva de la persona y reduciendo la brecha de desigualdad.

En resumen, este trabajo tiene un claro impacto en la mejora de la calidad educativa, la accesibilidad a la información y la promoción del desarrollo sostenible de acuerdo con los objetivos globales.

Bibliografía

[1] OECD, "Education at a glance 2025: OECD indicators," OECD Publishing, Tech. Rep., 2025.

[2] AENOR, "Une 153101:2018 ex: Lectura fácil. pautas y recomendaciones para la elaboración de documentos," Asociación Española de Normalización (AENOR), Madrid, 2018, norma española.

[3] Óscar García Muñoz, *LECTURA FÁCIL: MÉTODOS DE REDACCIÓN Y EVALUACIÓN*. Real Patronato sobre Discapacidad, 2012.

[4] N. Unidas, "Convención sobre los derechos de las personas con discapacidad," Organización de las Naciones Unidas, 2006.

[5] L. P. Morais, "Analizador de lectura fácil," Trabajo de Fin de Grado, Universidad Politécnica de Madrid, 2022, grado en Ingeniería Informática.

[6] D. D. Mladenova, "Lectura fácil: Oraciones coordinadas explicativas," Trabajo de Fin de Grado, Universidad Politécnica de Madrid, 2025, grado en Ingeniería Informática.

[7] R. R. Martín, "Servicio web para lectura fácil," Trabajo de Fin de Grado, Universidad Politécnica de Madrid, 2019, grado en Ingeniería Informática.

[8] O. Carreras, "Lectura fácil: pautas y recomendaciones," Feb. 2019, [Online]. Disponible: https://olgacarreras.blogspot.com/2019/02/lectura-facil-pautas-y-recomendaciones.html.

[9] D. M. Gamo and M. C. Suárez-Figueroa, *Razonamiento con imprecisión: lógica borrosa. Apuntes y ejercicios*. Madrid, Spain: Escuela Técnica Superior de Ingenieros Informáticos, Univ. Politécnica de Madrid, 2017.

[10] Python Software Foundation, "Python programming language," 2024, [Online]. Disponible: https://www.python.org/.

[11] Explosion AI, "spacy: Industrial-strength natural language processing," 2024, [Online]. Disponible: https://spacy.io/.

[12] J. Warner, J. Sexauer, and e. a. Van den Broeck, "JDWarner/scikit-fuzzy: Scikit-Fuzzy 0.5.0," Aug. 2024, dOI: 10.5281/zenodo.13372212. [Online]. Disponible: https://zenodo.org/records/13372212. Accedido: 2 de Nov. 2025.

[13] Artifex Software, “Pymupdf documentation,” 2024, [Online]. Disponible: https://pymupdf.readthedocs.io/en/latest/.

[14] Python Software Foundation, “python-pptx documentation,” 2024, [Online]. Disponible: https://python-pptx.readthedocs.io/en/latest/.

[15] ——, “python-docx documentation,” 2024, [Online]. Disponible: https://python-docx.readthedocs.io/en/latest/.

[16] ——, “Módulo os — interfaz miscelánea del sistema operativo,” 2024, [Online]. Disponible: https://docs.python.org/es/3.10/library/os.html.

[17] Pallets Project, “Flask documentation,” 2024, [Online]. Disponible: https://flask.palletsprojects.com/es/stable/.

[18] Microsoft Corporation, “Visual studio code,” 2024, [Online]. Disponible: https://code.visualstudio.com/.

[19] Real Academia Española, “Banco de datos (corpes xxi): Listado de frecuencias,” 2024, [Online]. Disponible: https://corpus.rae.es/lfrecuencias.html.

[20] KeepCoding, “¿qué es y en qué consiste la ley de zipf?” 2024, [Online]. Disponible: https://keepcoding.io/blog/que-es-y-en-que-consiste-la-ley-de-zipf/.

[21] Universal Dependencies contributors, “Universal dependencies,” 2024, [Online]. Disponible: https://universaldependencies.org/.

[22] Ontology Engineering Group, “Ontology engineering group - universidad politécnica de madrid,” 2025. [Online]. Available: https://oeg.fi.upm.es/

[23] Plena inclusión, “Plena inclusión - confederación española de organizaciones en favor de las personas con discapacidad intelectual o del desarrollo,” 2025. [Online]. Available: https://www.plenainclusion.org/

[24] Planeta Fácil, “Planeta fácil - noticias fáciles de entender,” 2025. [Online]. Available: https://planetafacil.plenainclusion.org/

[25] Sinpromi, “Biblioteca social educativa (base),” 2025. [Online]. Available: https://sinpromi.es/accesibilidad-universal/biblioteca-social-educativa-base/

[26] Plena inclusión Andalucía, *¿Qué es la Lectura fácil?*, Plena inclusión, 2021. [Online]. Available: https://www.plenainclusion.org/wp-content/uploads/2021/07/Plena-inclusion-Andalucia.-Que-es-la-lectura-facil.pdf

[27] O. Henry, *La última hoja*, 1st ed. Tenerife: Sociedad Insular para la Promoción de las Personas con Discapacidad (SINPROMI), dic 2014, adaptación de Esther Pulido del Río. Ilustraciones del Taller Giro Arte. Lectura Fácil.

[28] España, “Constitución española,” Boletín Oficial del Estado, n.º 311, dic 1978, ref. BOE-A-1978-31229. [Online]. Available: https://www.boe.es/eli/es/c/1978/12/27/(1)/con

[29] Sinpromi and Biblioteca Social Educativa, "Pensar en igualdad: guía en lectura fácil," Biblioteca BASE, s.f. [Online]. Available: https://bibliotecasocialeducativa.wordpress.com/publicaciones-en-lf/lectura-facil/

[30] J. Ortega y Gasset, *Ideas y creencias*. Biblioteca Virtual OMEGALFA, 2010, edición digital. [Online]. Available: https://www.omegalfa.es

[31] R. Velázquez Velázquez, "Unidad I: Introducción a la ciencia de la historia," Universidad Autónoma del Estado de Hidalgo, Escuela Preparatoria Número 1, 2015.

[32] R. Romero Ramírez, "Polibio de megalópolis," Universidad Veracruzana, Facultad de Historia, 2015, diapositivas de la asignatura Historiografía Grecolatina y Medieval.

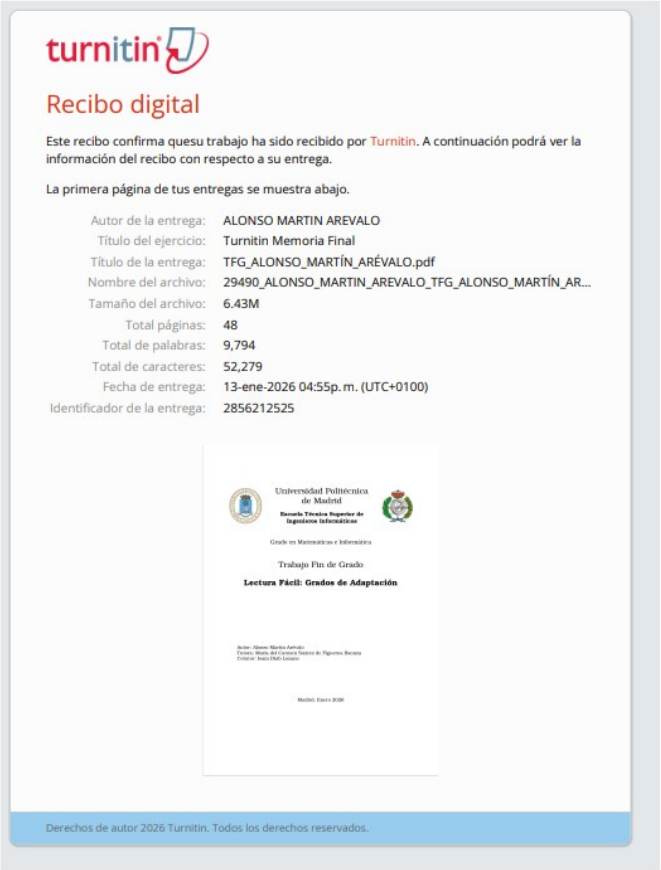
[33] Ministerio de la Presidencia, Relaciones con las Cortes e Igualdad, "Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público," *Boletín Oficial del Estado*, vol. 227, pp. 90 533–90 549, sep 2018, referencia: BOE-A-2018-12699. [Online]. Available: https://www.boe.es/eli/es/rd/2018/09/07/1112

[34] Parlamento Europeo y Consejo de la Unión Europea, "Directiva (UE) 2019/882, de 17 de abril de 2019, sobre los requisitos de accesibilidad de los productos y servicios," *Diario Oficial de la Unión Europea*, vol. L 151, pp. 70–115, jun 2019, referencia: DOUE-L-2019-80999.

[35] Naciones Unidas, "Objetivos y metas de desarrollo sostenible," s.f. [Online]. Available: https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/

Anexos

Informe de originalidad de Turnitin


turnitin

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	ALONSO MARTIN AREVALO
Título del ejercicio:	Turnitin Memoria Final
Título de la entrega:	TFG_ALONSO_MARTÍN_ARÉVALO.pdf
Nombre del archivo:	29490_ALONSO_MARTIN_AREVALO_TFG_ALONSO_MARTÍN_AR...
Tamaño del archivo:	6.43M
Total páginas:	48
Total de palabras:	9,794
Total de caracteres:	52,279
Fecha de entrega:	13-ene-2026 04:55p. m. (UTC+0100)
Identificador de la entrega:	2856212525

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

Lectura Fácil: Grados de Adaptación

Derechos de autor 2026 Turnitin. Todos los derechos reservados.

Figura 1: Recibo del informe de originalidad

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jan 14 14:21:53 CET 2026
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)