Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

LECTURA FÁCIL: FRASES NOMINALES y USOS NOMINALES de ADJETIVOS

Autor: Paula Castillo García

Tutor: María del Carmen Suárez de Figueroa Baonza

Cotutor: Isam Diab Lozano

Madrid, junio de 2025

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: FRASES NOMINALES y USOS NOMINALES de ADJETIVOS
Junio de 2025
Autor: Paula Castillo García

Tutor:
María del Carmen Suárez de Figueroa Baonza
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

El presente Trabajo de Fin de Grado tiene como propósito el diseño e implementación de un sistema automático capaz de identificar y adaptar adjetivos sustantivados en textos escritos en español, siguiendo los principios de la Metodología de Lectura Fácil o MLF. Estas construcciones gramaticales son habituales en el lenguaje formal y técnico, y pueden suponer un obstáculo importante para personas con dificultades lectoras o cognitivas. Por ello, su tratamiento automatizado representa un avance significativo en el ámbito de la accesibilidad textual

El sistema desarrollado combina reglas lingüísticas con modelos de lenguaje preentrenados, trabajando de forma modular en tres fases consecutivas. La primera se encarga de procesar un conjunto de frases proporcionadas como entrada para identificar estructuras gramaticales complejas. En la segunda fase, se aplica un conjunto de patrones definidos con la biblioteca de Procesamiento del Lenguaje Natural (NLP) spaCy, que permiten localizar adjetivos sustantivados, bien mediante determinantes, cuantificadores, contracciones y numerales, o bien mediante la partícula *'lo'*. Finalmente, en la tercera fase se procede a la adaptación de las frases detectadas utilizando estrategias distintas según el tipo de estructura. Los métodos de adaptación han consistido, o bien en la aplicación de un sistema de reglas; o bien en la utilización de modelos generativos de lenguaje: RoBERTa y Salamandra.

El sistema se ha evaluado sobre un corpus de más de 300 frases con resultados que demuestran su eficacia. A pesar de ello, a lo largo del desarrollo del proyecto se han identificado algunas limitaciones técnicas y lingüísticas que han servido como punto de partida para establecer futuras líneas de mejora.

Abstract

This Final Degree Project aims to design and implement an automatic system capable of identifying and adapting substantivized adjectives in Spanish texts, following the principles of the Easy-to-Read Methodology (MLF). These grammatical constructions are common in formal and technical language and can pose a significant barrier for people with reading or cognitive difficulties. Therefore, their automated processing represents a meaningful advancement in the field of textual accessibility.

The system developed combines linguistic rules with pre-trained language models and operates in a modular structure composed of three consecutive phases. The first phase processes a set of input sentences to identify complex grammatical structures. In the second phase, a series of patterns defined using the Natural Language Processing (NLP) library spaCy are applied to detect substantivized adjectives, either introduced by determiners, quantifiers, contractions and numerals, or by the particle *'lo'*. In the final phase, the detected phrases are adapted using different strategies depending on their structure. These adaptation methods consist either of applying a rule-based system or of using generative language models: RoBERTa and Salamandra.

The system was evaluated on a corpus of over 300 sentences, with results demonstrating its effectiveness. Nevertheless, during development, several technical and linguistic limitations were identified, which have served as a basis for defining future improvements.

Tabla de contenidos

1 Introducción

La lectura y comprensión de textos escritos representa una habilidad fundamental para el acceso al conocimiento y la comunicación entre personas. Sin embargo, una parte significativa de la población encuentra obstáculos en la comprensión de ciertos documentos debido a su complejidad léxica y sintáctica. En particular, personas con discapacidad intelectual, dificultades en el uso del idioma o con un bajo nivel de alfabetización son especialmente vulnerables a la exclusión informativa. Para abordar esta problemática, han surgido diversas metodologías que tratan de hacer la información más accesible, entre las que destaca la **Metodología de Lectura Fácil (MLF)**.

En la norma UNE 153101:2018 EX[1], la definición oficial de MLF es la siguiente:

> [1] *Método que recoge un conjunto de pautas y recomendaciones relativas a la redacción de textos al diseño y maquetación de documentos y a la validación de la comprensibilidad de los mismos, destinado a hacer accesible la información a las personas con dificultades de comprensión lectora.*

La MLF proporciona un conjunto de recomendaciones orientadas a mejorar la accesibilidad cognitiva de los textos. Entre estas recomendaciones se encuentran la eliminación de estructuras complejas, como los grupos nominales extensos o los adjetivos usados como sustantivos, que pueden dificultar la comprensión lectora. Estas pautas están recogidas en normas como la UNE[2] 153101:2018 EX y son respaldadas por asociaciones como *Inclusion Europe*[3] o *AENOR*.

El presente trabajo, se ha centrado en el estudio de los adjetivos sustantivados, es decir, aquellos adjetivos que, por el contexto gramatical, funcionan como sustantivos. Estas construcciones suelen omitir el sustantivo al que se refieren, exigiendo al lector un esfuerzo adicional de inferencia. Este tipo de elipsis, aunque frecuente en el lenguaje escrito, puede suponer una barrera para personas con dificultades lectoras, ya que reduce la concreción y la transparencia del mensaje.

A continuación, se presentan algunos ejemplos de frases que contienen adjetivos sustantivados y que podrían considerarse problemáticas según la MLF:

---

[1] Norma española que explica cómo hacer la lectura fácil.
[2] https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0060036#.WwVKBiC-mM-
[3] https://www.inclusion-europe.eu/easy-to-read-standards-guidelines/

"Los responsables serán sancionados." → AS[4]: "*responsables*".

"Ayudamos a los necesitados." → AS: "*necesitados*".

"Lo importante es participar." → AS: *"importante*".

"Todas las posibles fueron evaluadas." → AS: *"posibles*".

"Prefiero el azul al verde." → AS: *"azul"*, "*verde"*.

"Los buenos siempre ganan." → AS: *"buenos"*.

En todas estas frases, la información sustantiva queda implícita. Por ejemplo, *"los necesitados"* no especifica qué necesitan ni quiénes son; *"lo importante"* es una abstracción sin referente claro; y *"el azul"* exige deducir a qué objeto hace referencia. Este tipo de expresiones, aunque habituales, son precisamente las que este proyecto busca detectar y adaptar automáticamente para hacer los textos más comprensibles y accesibles.

En este contexto, el presente Trabajo de Fin de Grado tiene como objetivo el diseño y desarrollo de un sistema permita detectar automáticamente estos patrones lingüísticos complejos, y transformarlos siguiendo las recomendaciones de la Lectura Fácil. Para ello, se emplean técnicas de procesamiento del lenguaje natural (PLN) en lengua española, combinando análisis morfosintáctico con reglas lingüísticas y modelos de lenguaje preentrenados.

El sistema resultante está pensado para integrarse en una aplicación web de lectura fácil o como una prueba de concepto accesible en línea. A través de esta herramienta se espera facilitar la validación de las transformaciones propuestas y contribuir al desarrollo de soluciones tecnológicas orientadas a la accesibilidad cognitiva.

En el Anexo 3 de la presente Memoria, se encuentra adjunto el recibo del informe de originalidad de este trabajo, realizado mediante la herramienta Turnitin.

[4] Adjetivo sustantivado

2 Estado de la cuestión

En el campo del PLN, la adaptación automática de textos a Lectura Fácil ha sido un tema de creciente interés, particularmente en los últimos diez años. En la literatura se distinguen dos enfoques principales: por un lado, los trabajos que analizan si un texto cumple con las pautas de Lectura Fácil; y por otro, aquellos que transforman automáticamente los textos para ajustarlos a dichas pautas.

Dentro del primer grupo, destaca la herramienta ERAS[5], desarrollada por el Instituto de Ingeniería del Conocimiento (IIC) en colaboración con la Universidad Autónoma de Madrid (UAM). ERAS es un analizador automático que evalúa el grado de adecuación de un texto según las directrices de Lectura Fácil establecidas en la norma UNE 153101:2018 EX. Esta herramienta analiza diversos aspectos formales y lingüísticos del texto, como pueden ser la longitud media de las oraciones, la complejidad léxica, el uso de negaciones o estructuras pasivas, y elabora un informe detallado sobre su accesibilidad cognitiva. Sin embargo, su propósito es diagnóstico, no transformador, por lo que no realiza adaptaciones del texto de forma automática. [2]

En el segundo enfoque, la investigación se ha centrado tradicionalmente en la simplificación de textos (*text simplification*), que abarca tanto la simplificación léxica —sustitución de términos complejos por otros más accesibles— como la simplificación sintáctica, es decir, la reestructuración de frases para hacerlas más comprensibles. Uno de los sistemas más relevantes en este ámbito ha sido Simplext[6], desarrollado en el Centro de Inteligencia Digital (CENID) de la Universidad de Alicante (UA), que combina análisis lingüístico y generación de texto para adaptar adaptar textos en diferentes niveles de dificultad, con especial foco en la inclusión de personas con discapacidad o dificultades cognitivas. [3]

Posteriormente, se desarrolló EASIER[7], dentro del grupo *Human Language and Accessibility Technologies* (HULAT) del Departamento de Informática de la Universidad Carlos III de Madrid (UC3M). Esta es una plataforma en línea que asiste a personas con dificultades lectoras mediante funcionalidades como la simplificación automática, definiciones contextuales y reformulaciones accesibles. El sistema integra técnicas de procesamiento lingüístico con un enfoque centrado en la accesibilidad cognitiva. [4]

Más recientemente, se han desarrollado soluciones orientadas a un uso práctico por parte de instituciones y usuarios no especializados. Es el caso de **ATECA**[8], una aplicación creada por el Centro Español de Accesibilidad Cognitiva (CEACOG) en colaboración con la Universidad Politécnica de Madrid (UPM), que permite adaptar textos al formato de Lectura Fácil mediante una interfaz sencilla, diseñada para entornos educativos, sociales y sanitarios. [5]

---

[5] Evaluador de Reglas para la Accesibilidad del texto escrito en español
[6] https://simpletext.demos.gplsi.es/
[7] easier
[8] ATECA - Inicio de Sesión

Con la aparición de modelos de lenguaje avanzados basados en arquitecturas Transformer (Vaswani et al., 2017), se ha intensificado el uso de modelos preentrenados como BERT [6] o RoBERTa [7] en tareas de simplificación automática. Aunque estas tecnologías han mostrado buenos resultados en términos de fluidez y coherencia, suelen carecer de entrenamiento específico en los principios de Lectura Fácil, por lo que sus salidas requieren evaluación adicional para garantizar la accesibilidad real del contenido generado.

Finalmente, cabe destacar que la mayoría de estos sistemas no abordan de forma específica la transformación de adjetivos sustantivados ni los grupos nominales complejos. Por tanto, este TFG se sitúa en un área poco explorada hasta la fecha, proponiendo un enfoque novedoso centrado en estos fenómenos lingüísticos concretos, esenciales para mejorar la comprensibilidad de los textos según la MLF.

3 Metodología

Este capítulo describe el enfoque seguido para el desarrollo del sistema de identificación y adaptación de adjetivos sustantivados. El trabajo se ha organizado en tres fases principales: identificación de estructuras problemáticas en el texto, clasificación de las frases según el tipo de sustantivación, y generación automática de adaptaciones accesibles.

La metodología combina técnicas lingüísticas basadas en patrones con modelos de lenguaje preentrenados, lo que permite un tratamiento robusto y flexible de los distintos casos. Además, se detallan las herramientas tecnológicas utilizadas, el entorno de desarrollo, y el proceso de adaptación del código base sobre el que se ha construido el sistema final.

3.1 Arquitectura general del sistema

Para tener una visión general y elevada del proyecto, se ha realizado un diagrama de flujo[9] en el que se indica el funcionamiento del sistema. La Figura 1 muestra que el programa pasa por tres fases, primero se recoge el conjunto de frases que se desean adaptar, posteriormente pasan por una fase de identificación, de la que resultan dos archivos según la clasificación identificada. Por último, se lleva a cabo la fase de adaptación en la que se reescriben, por separado, los dos archivos generados por la fase anterior y se genera uno nuevo por cada uno de ellos.

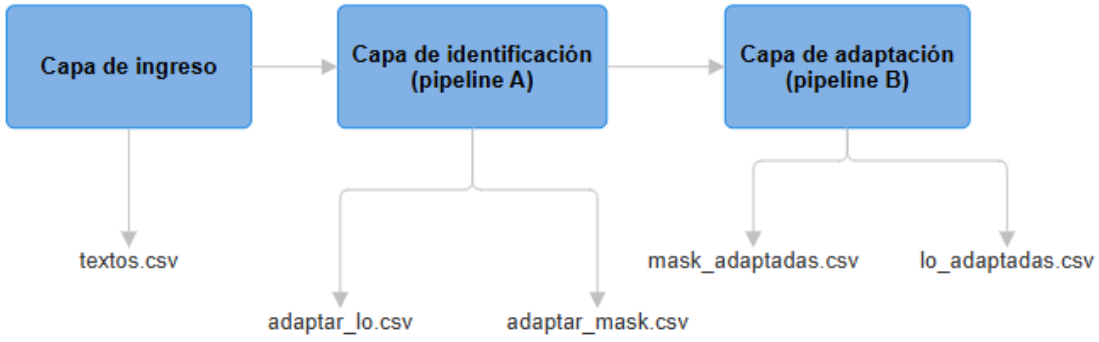

Figura 1: Diagrama de flujo de la arquitectura del sistema

[9] El diagrama ha sido elaborado con la herramienta SmartDraw: https://www.smartdraw.com/flowchart/diagramas-de-flujo.htm?srsltid=AfmBOoqPlSPXEmhURHmFxiuSjWb6LTiYRCPirMgDIUXaaJzKPjC5SKs-

3.2 Herramientas y tecnologías utilizadas

Para la elaboración del presente proyecto, ha sido fundamental el uso de diversas herramientas tecnológicas, tras su previo estudio y familiarización. A continuación, se van a describir estos elementos, junto con la utilidad que se les ha dado en el código.

3.2.1 Lenguajes y librerías

La codificación se ha sido realizada con **Python**, un lenguaje de programación de alto nivel, poderoso y flexible, con múltiples ventajas. Destaca la simplicidad y claridad de su sintaxis, lo cual ha resultado muy útil en la elaboración de este proyecto. Permite prototipar rápidamente y dispone de un gestor de paquetes (pip) que facilita la instalación de dependencias.

Esta herramienta también agilizaría la integración del código con Front-End, debido a su versatilidad en el campo del desarrollo web. Además, Python cuenta con una amplia biblioteca estándar y ofrece la posibilidad de utilizar librerías para análisis de datos y *Machine Learning* (ML) que han resultado fundamentales para el desarrollo del programa. [8]

Dichas librerías son las siguientes:

spaCy: Framework de NLP en Python, optimizado para producción. Está diseñada para ser rápida, eficiente y fácil de usar, especialmente para tareas del mundo real que involucran grandes volúmenes de texto. Cuenta con soporte para más de 75 idiomas, 84 pipes entrenados para 25 idiomas y vectores de palabras preentrenados. [9]
En el código se ha hecho uso de dos modelos de spaCy: ***es_dep_news_trf*** (en la identificación de patrones sintácticos) y ***es_core_news_md*** (en la adaptación de frases). El primero es el más adecuado para el análisis sintáctico y morfológico de alta precisión, ya que está basado en **transformers**; especialmente útil para *parsing*[10] y etiquetas gramaticales, aunque a costa de más memoria/CPU y sin NER (reconocimiento de entidades). Mientras, el segundo modelo es ideal para identificar similitudes semánticas y tareas que requieran representaciones léxicas, gracias a su etiquetado ligero, lematización y vectores *word-embeddings* [11] ; tiene un rápido análisis en CPU, y cuenta con reconocimiento de entidades. [10]

[10] Proceso de analizar un texto o código para entender su estructura y extraer información significativa.
[11] Métodos para representar las palabras de un texto mediante vectores numéricos reales.

Además de los modelos, otro elemento esencial ha sido el **Matcher**, un componente de spaCy que permite detectar patrones lingüísticos en un objeto Doc de forma declarativa, sin necesidad de escribir expresiones regulares complejas sobre texto crudo. Para ello, hace uso de reglas que describen sus atributos de *token* (como las etiquetas de texto o de categorías gramaticales). Gracias al Matcher se ha alcanzado una detección muy precisa y robusta de estructuras gramaticales, apoyándose en el análisis sintáctico y morfológico realizado previamente por spaCy. [11]

HuggingFace Transformers: Biblioteca que permite trabajar con modelos preentrenados de PLN, visión artificial, audio y multimodales para inferencia y entrenamiento. [12] Los modelos a los que provee acceso son de última generación, en concreto para el desarrollo del proyecto se ha utilizado una versión del modelo **RoBERTa**: *PlanTL-GOB-ES/roberta-base-bne*. Esta versión, sigue la misma estructura que el modelo roberta-base, pero ha sido adaptada y entrenada exclusivamente con datos en español. Para su pre-entrenamiento se han usado 570 GB de texto limpio y deduplicado obtenido de los rastreos web realizados por la Biblioteca Nacional de España entre 2009 y 2019, lo que le confiere un amplio conocimiento del uso real del español en distintas fuentes y registros. [13]

Lo útil y relevante para el tema que nos ocupa, es que este modelo se especializa en *Masked Language Modeling* (MLM), es decir, rellena automáticamente tokens ocultos en una secuencia de texto (*fill-mask*). Esto ha sido fundamental en la parte de adaptación de las frases mediante un modelo generativo como alternativa a las reglas. No obstante, al entrenarse sobre datos web, puede reflejar sesgos presentes en dichas fuentes, por lo que conviene evaluar las predicciones en contextos sensibles.

Otro modelo perteneciente a esta biblioteca y que ha sido de alta relevancia en el proyecto es el modelo **Salamandra**: BSC-LT/salamandra-7b-instruct.
Salamandra-7B es un modelo generativo capaz de producir respuestas contextualmente coherentes a partir de indicaciones expresadas en lenguaje natural. Ha sido entrenado con una mezcla de corpus generalistas y especializados, incluyendo textos técnicos, educativos y administrativos, lo que le permite manejar bien construcciones abstractas, definiciones y reformulaciones. En este proyecto, se ha utilizado para generar reformulaciones más explícitas de construcciones abstractas del tipo *'lo'* + adj, solicitándole, mediante *prompts*, que reescriba dichas frases eliminando el pronombre *'lo'* sin alterar su significado. [14]

PyTorch (torch): Una de las librerías líderes en Python, elaborada para cómputo numérico y aprendizaje profundo (*deep learning*). La usa por defecto HuggingFace Transformers como motor de cómputo tensorial; carga y ejecuta sus modelos cuando se dispone de GPU, permitiendo acelerar el llenado de máscaras. [15]

configparser: Módulo de la biblioteca estándar de Python que permite leer y escribir ficheros de configuración en formato INI, organizados en secciones y pares clave–valor. [16] Esta herramienta ha sido de gran ayuda para la organización del código y la definición del acceso a elementos esenciales, como los modelos de spaCy o el umbral de similitud vectorial.

3.2.2 Entorno de desarrollo

La organización y visualización del código ha constituido una parte esencial en el desarrollo del mismo. Se ha hecho uso de herramientas que han ayudado a agilizar el trabajo, reduciendo posibles demoras debidas a una mala disposición de la información, así como pérdidas o superposiciones de versiones del código.

El proyecto se ha desarrollado principalmente en **Visual Studio Code**, un editor de texto ligero y extensible que facilita la navegación de carpetas, la gestión de *snippets* y la integración con terminales y control de versiones. [17]

El control de versiones se ha gestionado con Git, usando un repositorio remoto en **GitLab**. Para ello se han creado ramas de características y se ha empleado un flujo básico de git add → git commit → git push con mensajes descriptivos. El repositorio[12] incluye todos los archivos necesarios para la ejecución del sistema.

Para aislar dependencias de Python se ha usado un **entorno virtual** (venv). En este entorno se han instalado todas las librerías necesarias, para así evitar conflictos con paquetes del sistema y facilitar la reproducibilidad.

Este conjunto de herramientas ha proporcionado un flujo de trabajo ágil, seguro y replicable, permitiendo concentrarse en el desarrollo de los pipelines de NLP sin interferencias externas.

3.3 Origen del código base y adaptación de implementaciones previas

El código resultante de este proyecto ha sido el fruto de la adaptación de un código base proporcionado por los tutores del presente trabajo. Dicho código fue desarrollado por Marina Pindado, integrante de la línea de investigación "Inteligencia Artificial Inclusiva y Aplicada: Mejora de la Accesibilidad Cognitiva".

Se trata de un notebook de Python que contaba con la estructura base para la identificación y adaptación de textos. Además, incluye títulos explicativos para cada parte del código, lo cual resultó de gran ayuda para comprender el tema al inicio del proyecto. La idea de separar la adaptación entre los dos tipos de estructuras se incluía en esta implementación previa, aunque no realizaba estas tareas correctamente. La adaptación del código base ha consistido en la ampliación y corrección de patrones de la fase de detección, la cual estaba bastante avanzada, aunque también han sido necesarias numerables modificaciones. Respecto a la parte de adaptación, ha sido prácticamente reescrita de nuevo, ya que se ha cambiado el enfoque original del código base, incluyendo la adaptación por reglas y modelos generativos más concretos y adecuados.

[12] https://gitlab.com/paulacastillog05/tfg-lectura-facil

4 Desarrollo e implementación

El grueso de este proyecto ha consistido en el desarrollo del código que implementa las funcionalidades para los objetivos establecidos. Por ello, en esta codificación reside toda la responsabilidad de la correcta actuación del programa.

Para la profunda comprensión del código, se va a realizar una descripción detallada y justificada de cada una de sus partes. Además, se va a proceder a la explicación de las pruebas realizadas y el análisis de la validación de los resultados obtenidos.

La estructura general del repositorio se puede consultar en el Anexo 1 de esta Memoria. Esta proporciona una vista elevada del proyecto y explica la funcionalidad de cada módulo.

4.1 Fase 1: Identificación

Esta parte del código se encarga de recorrer el conjunto de frases y analizarlas utilizando el modelo lingüístico de spaCy. El objetivo es identificar aquellas estructuras que puedan considerarse problemáticas según la MLF. Para ello, se realiza un análisis gramatical basado en la tarea de etiquetado morfosintáctico (PoST[13]), que permite asignar a cada palabra su categoría gramatical y función dentro de la oración. A partir de estas etiquetas, el sistema aplica un conjunto de patrones que detectan adjetivos sustantivados y clasifica las frases en dos grupos, según el tipo de estructura identificado: aquellas marcadas con <mask> y aquellas que contienen la construcción *'lo'* + adjetivo.

El archivo de entrada es un csv con frases, del que se hablará más en profundidad en el apartado de pruebas. La tarea del detector será saber cuáles necesitan una adaptación, y en caso afirmativo, de qué tipo de estructura se trata.

El código descrito a continuación forma parte del archivo *detection.py*.

En primer lugar, como se puede observar en la Figura 2, se importan todas las librerías necesarias, así como elementos de la configuración del proyecto.

```
1  import spacy
2  from spacy.language import Language
3  from spacy.matcher import Matcher
4  from spacy.tokens import Token
5  from spacy.symbols import ORTH
6  from .pipeline_base import BasePipeline
7  from .utils import guardar_csv, insertar_mask
8  import csv
9  from pathlib import Path
10 from nlp import Config
11 cfg = Config()
```

Figura 2: Importación de librerías necesarias para detección

[13] *Part-of-Speech tagging*

A continuación, para llevar a cabo la detección de adjetivos sustantivados, se ha desarrollado una clase denominada *SustantivadosPipeline*, que extiende de una clase base común a otros componentes del proyecto. Esta clase se encarga de cargar el modelo lingüístico de spaCy especificado en la configuración, procesar los textos y aplicar un componente personalizado que detecta patrones lingüísticos compatibles con la sustantivación de adjetivos. Véase la Figura 3, donde se realiza la carga inicial de la clase.

```
def load(self):
    self.nlp = spacy.load(self.model_name)
    Token.set_extension('es_sustantivado', default=False, force=True)
    # caso especial para <mask>
    special = [{ORTH: '<mask>'}]
    self.nlp.tokenizer.add_special_case('<mask>', special)
    self.nlp.add_pipe('sustantivados_component', last=True)
```

Figura 3: Definición de la carga inicial de la clase *SustantivadosPipeline*

El componente principal se añade al pipeline de spaCy mediante la función sustantivados_component, registrada con el decorador @Language.component. Este componente realiza tres tareas clave, las cuales se van a exponer a continuación.

4.1.1 Corrección de etiquetas Part-of-Speech (PoS)

Antes de proceder a la identificación de las estructuras problemáticas, ha sido necesaria la corrección de algunos de los etiquetados realizados por SpaCy (Véase Figura 4).

- Se reclasifican ciertos adjetivos como sustantivos si aparecen como objetos directos o indirectos de un verbo. Esto se detectó en una de las pruebas con la frase: *"¿Qué vestido te gusta más? – El verde es el más bonito."*, en la que el etiquetado no se realizó correctamente, ya que no identificó *"vestido"* como un NOUN (sustantivo), si no como un ADJ (adjetivo).

- Asimismo, si un sustantivo tiene un lema que corresponde a un color o una nacionalidad (según unas listas predefinidas en la configuración), se etiqueta como adjetivo. Esto también se detectó mediante uno de los textos de prueba: *"El marrón te queda mejor, aunque el negro combine más."*, en la cual *"negro"* se etiquetó como NOUN (sustantivo) en lugar de ADJ (adjetivo).

Estas modificaciones son esenciales ya que el etiquetado conforma la base para la posterior clasificación y adaptación.

```python
# Reclasificación de ADJ con función de objeto de verbo
if token.pos_ == "ADJ" and token.dep_ in ("obj", "iobj") and token.head.pos_ == "VERB":
    token.pos_ = "NOUN"
# Reclasificación de colores y nacionalidades clasificados como NOUN
if token.pos_ == "NOUN" and (token.lemma_.lower() in cfg.colores or token.lemma_.lower() in cfg.nacionalidades):
    print(token.text)
    token.pos_ = "ADJ"
```

Figura 4: Corrección de etiquetado de SpaCy

Inicialmente, se pensó en identificar las nacionalidades mediante la característica de entidad del token (token.ent_type_). Esta es una cadena de texto que indica el tipo de entidad nombrada al que pertenece ese token, de acuerdo con el componente de *"NER"* (*Named Entity Recognition*). En este caso se relacionó con la etiqueta *"NORP"* (*Nationalities or religious or political groups*). Sin embargo, durante las pruebas se comprobó que esto no cubría los casos, ya que el modelo de SpaCy elegido, al ser un modelo español, sólo usa cuatro entidades, entre las cuales no se encuentra *"NORP"*. Por ello se optó por tratarlo de igual manera que los colores. Así, desde el archivo de configuración se pueden modificar fácilmente tanto la lista de colores como la de nacionalidades, según se quieran añadir o retirar palabras.

4.1.2 Identifiación de patrones con spaCy Matcher

A continuación, para identificar las estructuras problemáticas se ha implementado un conjunto de patrones lingüísticos mediante el componente Matcher de SpaCy. Este componente permite definir secuencias de tokens que cumplan ciertas condiciones gramaticales (como la categoría léxica, la dependencia sintáctica, o la forma léxica) y aplicarlas a un documento procesado para identificar coincidencias.

Los patrones definidos en este proyecto no se limitan a casos básicos, si no que cubren una amplia variedad de construcciones en las que puede aparecer un adjetivo sustantivado. La mayoría de los patrones, a excepción de un caso que se expondrá más adelante, comparten una característica común: buscan un adjetivo (ADJ) con una función sintáctica específica precedido por una combinación que active la sustantivación.

En primer lugar, se han definido patrones de determinación directa. Estos patrones detectan secuencias en las que el adjetivo sustantivado está precedido por un determinante, cuantificador, numeral o contracción. Todos ellos contemplan además un adverbio opcional entre el determinante y el adjetivo, lo cual permite capturar estructuras muy frecuentes como *"los más altos"*, *"al menos indicado"*, o *"todas las menos preparadas"*. La condición adverbial se implementa mediante el operador "OP": "?".

Los adjetivos son filtrados no solo por su categoría POS (ADJ), sino también por su función sintáctica (DEP), restringida a un conjunto de dependencias gramaticales relevantes para sustantivación:

nsubj (sujeto nominal)

obj (objeto directo)

iobj (objeto indirecto)

appos (aposición)

conj (coordinación)

ROOT (raíz)

obl (complemento oblicuo)

Los patrones definidos permiten identificar adjetivos sustantivados precedidos de:

(1) Cuantificadores (POS: CUAN) como *'alguno'*, *'ninguno'*, *'todo'*.

Ejemplo: *"todo ignorante"*.

(2) Determinantes (POS: DET) como *'el'*, *'la'*, *'una'*, *'los'*.

Ejemplo: "*el rojo*".

(3) Contracciones *'al'*, *'del'*.

Ejemplo: "*al guapo*".

(4) Numerales (POS: NUM) como *'diez', 'segunda'.*

Ejemplo; "*dos muy tontos*".

Por otra parte, se han incluido patrones con estructuras partitivo-genitivas ('de'), para cubrir casos menos comunes, pero igualmente relevantes como: *"Los de siempre"*.

Aunque este patrón no incluya la participación de un adjetivo, se han tratado estos casos de igual manera que los adjetivos sustantivados por su equivalencia sintáctica.

En estos patrones, un elemento antecedente (como *'el'*, *'muchos'*, *'tres'*) introduce un sintagma partitivo[14] o genitivo[15]. El término final puede ser un sustantivo (NOUN), nombre propio (PROPN), pronombre (PRON), adjetivo (ADJ) o adverbio (ADV), cubriendo así múltiples escenarios reales de uso:

(5) Determinante + *de* + término.

Ejemplo: *"la de verde"*.

(6) Cuantificador + *de* + término.

Ejemplo: *"muchos de ellos"*.

(7) Contracción (*al, del*) + *de* + término.

Ejemplo: *"al de siempre"*.

(8) Numeral + *de* + término.

Ejemplo: *"tres de los mejores"*.

[14] Que indica una parte de un todo.
[15] Que indica la posesión de algo.

Finalmente, se incluye el patrón para la estructura **(9) *"lo"* + adjetivo**, con un adverbio opcional intermedio. Esta construcción es una de las formas más frecuentes y claras de sustantivación en español, como en *"lo importante"*, *"lo más lógico"*, *"lo desconocido"*.

Este patrón es fundamental, ya que estas construcciones tienen un alto grado de abstracción semántica y representan uno de los principales objetivos de adaptación dentro de la MLF.

Todos los patrones son añadidos al Matcher mediante la instrucción matcher.add(...), lo que permite aplicar de forma eficiente la lógica definida sobre cada documento procesado.

En la Figura 5, que se muestra a continuación, se puede ver claramente cómo se definieron cada uno de estos patrones, siguiendo las especificaciones recién explicadas.

```python
# Creación de un matcher para identificar adjetivos sustantivados
matcher = Matcher(doc_t.vocab)

# Cuantificadores completos + Adverbio opcional + Adjetivo + Condición de dependencia
pattern_cuant = [{"POS": "CUAN" },
 {"POS": "ADV", "OP": "?"},
 {"POS": "ADJ", "DEP": {"IN": ['nsubj', 'iobj', 'obj', 'appos', 'conj', 'ROOT', 'obl']}}]

# Determinantes como "los", "las", etc. + Adverbio opcional + Adjetivo + Condición de dependencia
pattern_det = [{"POS": "DET" },
 {"POS": "ADV", "OP": "?"},
 {"POS": "ADJ", "DEP": {"IN": ['nsubj', 'iobj', 'obj', 'appos', 'conj', 'ROOT', 'obl']}}]

# Contracciones "al" y "del" + Adverbio opcional + Adjetivo + Condición de dependencia
pattern_contr = [{"LOWER": {"IN": ["al", "del"]}},
 {"POS": "ADV", "OP": "?"},
 {"POS": "ADJ", "DEP": {"IN": ['nsubj', 'iobj', 'obj', 'appos', 'conj', 'ROOT', 'obl']}}]

# Numerales + Adverbio opcional + Adjetivo + Condición de dependencia
pattern_num = [{"POS": "NUM"},
 {"POS": "ADV", "OP": "?"},
 {"POS": "ADJ", "DEP": {"IN": ['nsubj', 'iobj', 'obj', 'appos', 'conj', 'ROOT', 'obl']}}]

# Determinante + "de" + Numeral opcional + término (NOUN/PROPN/ADV/PRON)
pattern_de_det = [{"POS": "DET" },
    {"LOWER": "de"},
    {"POS": "NUM", "OP": "?"},
    {"POS": {"IN": ["NOUN", "PROPN", "ADV", "PRON", "ADJ"]}}]

# Cuantificador + "de" + Numeral opcional + término (NOUN/PROPN/ADV/PRON)
pattern_de_cuan = [{"POS": "CUAN" },
    {"LOWER": "de"},
    {"POS": "NUM", "OP": "?"},
    {"POS": {"IN": ["NOUN", "PROPN", "ADV", "PRON", "ADJ"]}}]
# Cuantificador + "de" + Numeral opcional + término (NOUN/PROPN/ADV/PRON)
pattern_de_cuan = [{"POS": "CUAN" },
    {"LOWER": "de"},
    {"POS": "NUM", "OP": "?"},
    {"POS": {"IN": ["NOUN", "PROPN", "ADV", "PRON", "ADJ"]}}]

# Contracción + "de" + Numeral opcional + término (NOUN/PROPN/ADV/PRON)
pattern_de_contr = [{"LOWER": {"IN": ["al", "del"]}},
    {"LOWER": "de"},
    {"POS": "NUM", "OP": "?"},
    {"POS": {"IN": ["NOUN", "PROPN", "ADV", "PRON", "ADJ"]}}]

# Numeral + "de" + término (NOUN/PROPN/ADV/PRON)
pattern_de_num = [{"POS": "NUM" },
    {"LOWER": "de"},
    {"POS": {"IN": ["NOUN", "PROPN", "ADV", "PRON", "ADJ"]}}
]

# 'lo'. + Adverbio opcional + Adjetivo + Condición de dependencia
pattern_lo = [{"LOWER": "lo"},
 {"POS": "ADV", "OP": "?"},
 {"POS": "ADJ", "DEP": {"IN": ['nsubj', 'iobj', 'obj', 'appos', 'conj', 'ROOT', 'obl']}}]

matcher.add("ADJ_SUST_CUANT", [pattern_cuant])
matcher.add("ADJ_SUST_DET", [pattern_det])
matcher.add("ADJ_SUST_CONTR", [pattern_contr])
matcher.add("ADJ_SUT_NUM", [pattern_num])
matcher.add("DET_DE_TERM", [pattern_de_det])
matcher.add("CUAN_DE_TERM", [pattern_de_cuan])
matcher.add("NUM_DE_TERM", [pattern_de_num])
matcher.add("CONTR_DE_TERM", [pattern_de_contr])
matcher.add("ADJ_SUST_LO", [pattern_lo])
```

Figura 5: Definición de patrones con Matcher de SpaCy

4.1.3 Marcado y almacenamiento de resultados

Una vez definidos los patrones, el Matcher se aplica al documento, devolviendo una lista de coincidencias (*matches*). Cada coincidencia contiene el identificador del patrón, así como las posiciones de inicio y fin del fragmento coincidente. El desarrollo de la clasificación de patrones, el cual se va a explicar a continuación, se puede ver en la Figura 6.

Para marcar los adjetivos sustantivados, se ha añadido a los *tokens* una extensión personalizada, denominada 'es_sustantivado'. Véase en la Figura 3, donde se introduce esta extensión con default=False, de manera que todos los tokens se inicializan como *no sustantivados*.

Cuando se recorre la lista de coincidencias, el adjetivo contenido en el fragmento coincidente se marca como sustantivado (es_sustantivado = True) y la frase se trata según el tipo:

- Si la estructura del fragmento coincide con el patrón **(9)**, se guarda la oración en el archivo *adaptar_lo.csv* para su posterior adaptación.

- En otro caso, se inserta <mask> en el lugar correspondiente[16] y se guarda la frase en el archivo *adaptar_mask*, para su posterior adaptación.

  Para este proceso se utiliza la función *insertar_mask* (véase Figura 7), que genera una versión modificada del texto original, insertando la cadena especial <mask> antes de cada adjetivo sustantivado. Consiste en recorrer la frase en busca de los tokens que hayan sido marcados como sustantivados. Si el token encontrado va precedido de un adverbio, la máscara se inserta antes del adverbio en lugar del adjetivo. Esto permite conservar el contexto completo en construcciones como *"los más altos"*, donde la expresión *"más altos"* actúa como un bloque y debe ser tratado como una unidad.
  Por último, se construye un nuevo texto token a token, insertando <mask> justo antes de los tokens marcados (o sus adverbios previos, en su caso), respetando los espacios originales del documento.

  El resultado es una versión del texto donde cada adjetivo sustantivado aparece precedido por la etiqueta <mask>, facilitando su localización para los módulos posteriores del sistema encargados de adaptar estos fragmentos.

A continuación, se muestra el código correspondiente a lo recién expuesto.

[16] el tratamiento de los patrones (1), (2), (3), (4) difiere del de los patrones (5), (6), (7), (8) en el token marcado como sustantivado. En el primer grupo se marca la última palabra del fragmento (que corresponde al adjetivo sustantivado), mientras que en el segundo caso se marca el penúltimo token del fragmento (el correspondiente a la palabra *'de'*, para asegurar la correcta inserción de <mask> posterior).

```python
# Aplicación del matcher a la fila
matches = matcher(doc_t)

hay_mask = False
ya_guardado_lo = False

for match_id, start, end in matches:
    pattern_name = doc_t.vocab.strings[match_id]
    span = doc_t[start:end]

    if pattern_name in ["ADJ_SUST_CUANT", "ADJ_SUST_DET", "ADJ_SUT_NUM", "ADJ_SUST_CONTR"]:
        span[-1]._.es_sustantivado = True
        hay_mask = True
    elif pattern_name in ["DET_DE_TERM", "CUAN_DE_TERM", "NUM_DE_TERM", "CONTR_DE_TERM"]:
        span[-2]._.es_sustantivado = True
        hay_mask = True
    elif pattern_name == "ADJ_SUST_LO" and not ya_guardado_lo:
        span[-1]._.es_sustantivado = True
        guardar_csv("adaptar_lo.csv", doc_t)
        ya_guardado_lo = True

# Solo guarda una vez en adaptar_mask.csv si hay al menos un sustantivado
if hay_mask:
    texto_con_mask = insertar_mask(doc_t)
    guardar_csv("adaptar_mask.csv", texto_con_mask)
```

Figura 6: Clasificación de patrones según el matcher

```python
# Función para insertar "[MASK]" delante de los tokens marcados
def insertar_mask(doc):
    mask_indices = set()
    for i, token in enumerate(doc):
        if token._.es_sustantivado:
            # Si hay un adverbio antes, se inserta la máscara antes de éste
            if i > 0 and doc[i-1].pos_ == "ADV":
                mask_indices.add(i-1)
            else:
                mask_indices.add(i)

    nuevo_texto = ""
    for i, token in enumerate(doc):
        if i in mask_indices:
            nuevo_texto += "<mask> " + token.text_with_ws
        else:
            nuevo_texto += token.text_with_ws
    return nuevo_texto
```

Figura 7: Inserción de <mask>

4.2 Fase 2: Adaptación

Una vez generados ambos archivos (*adaptar_mask.csv* y *adaptar_lo.csv*), producto de la identifiación y clasificación de las frases, el siguiente paso es su adaptación según la MLF.

Para llevar a cabo esta acción, se ha desarrollado una clase denominada *AdaptationPipeline*, que extiende de una clase base común a otros componentes del proyecto. Esta se encarga de adaptar los textos ya procesados y clasificados en la fase anterior, mediante distintos métodos según las necesidades de cada tipo de problemática.

Para la comprensión de esta fase, se ha dividido esta sección de la Memoria en dos grandes subapartados, donde se explicará detalladamente la adaptación de los dos tipos de frases identificadas en la fase de detección.

4.2.1 Caso 1: Adaptación de <mask>

Como se ha mencionado en apartados anteriores, esta tarea denominada *fill-mask*, consiste en la sustitución del token <mask> por una palabra, en este caso un sustantivo.

Este proceso tiene como objetivo transformar expresiones potencialmente abstractas en formulaciones más explícitas y comprensibles, facilitando así su lectura y comprensión. La idea ha sido proponer dos adaptaciones alternativas, una basada en reglas y otra mediante un modelo de inteligencia artificial.

La estructura encargada de esta tarea implementa el método *process_file_mask*, que recibe como entrada un archivo csv con frases marcadas, y produce un nuevo csv con tres columnas: la frase original, una adaptación generada por reglas lingüísticas, y otra generada mediante un modelo de lenguaje (RoBERTa). A continuación, se describe detalladamente cómo se realiza cada parte de esta adaptación, lo cual puede ser consultado de forma paralela en la Figura 8.

1. Localización de máscaras

Se recorre cada frase del archivo de entrada, buscando los tokens que contienen la etiqueta <mask>. Cada aparición se guarda en una lista ordenada que se procesará individualmente. Esto es crucial, ya que hay frases que pueden contener más de una etiqueta, por lo que es necesario tratar de forma independiente cada aparición, como se verá más adelante.

2. Identificación del adjetivo sustantivado

A partir de la posición de la máscara, se determina el adjetivo al que esta hace referencia. En caso de que exista un adverbio intermedio (por ejemplo, *"más inteligente"*, *"menos capacitado"*), se ajusta el índice para identificar correctamente el adjetivo principal que ha sido sustantivado.

3. Recuperación de información gramatical

Para proponer una adaptación coherente, se recupera del texto el determinante que acompaña al adjetivo sustantivado, ya que suele contener información de género y número.

En un primer momento la información morfológica se obtenía del adjetivo sustantivado, pero tras varias pruebas se encontró que muchos de estos sustantivos no tenían género y/o número, mientras que el determinante que los precede sí cuenta con estos rasgos en la mayoría de los casos. Por ejemplo, el adjetivo *"fuerte"* no proporciona información sobre el género, pero si miramos el determinante que lo sustantiva *('el' / 'la'*) podemos obtener los datos deseados. Estos datos son esenciales para seleccionar un sustantivo adecuado que concuerde gramaticalmente con el contexto.

```
# búsqueda del token <mask>
for i, token in enumerate(doc):
    if token.text == self.mask:
        masks.add(i)

mask_list = sorted(masks)

for n_mask, j in enumerate(mask_list, start=1):
    # búsqueda del adjetivo sustantivado
    adj = doc[j+2] if doc[j+1].pos_ == "ADV" else doc[j+1]
    text_low = adj.text.lower()

    # búsqueda del determinante que tenga información de género y número
    for k in range(j - 1, -1, -1):
        if(doc[k].pos_ == "DET"):
            det_genero = doc[k].morph.get("Gender")
            det_numero = doc[k].morph.get("Number")
            break

    genero = adj.morph.get("Gender")
    numero = adj.morph.get("Number")
```

Figura 8: Pasos previos a la adaptación

4.2.1.1 Adaptación basada en reglas lingüísticas

El sistema intenta generar una adaptación explícita, emprende una búsqueda del candidato adecuado para la sustitución, utilizando un conjunto jerárquico de reglas:

- **Casos de colores y nacionalidades**: Si el adjetivo es un color o una nacionalidad (según unas listas predefinida), se sustituye directamente por la palabra "color", en el caso de los colores; o por *"hombre"*, *"hombres"*, *"mujer"* o *"mujeres"* en el caso de las nacionalidades, según el género y número del adjetivo. (Véase Figura 10)

- **Sustantivos del contexto**: Si no se ha encontrado una adaptación anterior, se busca un sustantivo cercano en la frase que coincida en género y número. Esto se hace recorriendo los tokens del documento de entrada y comparando el género y número de cada sustantivo que encuentra. (Véase Figura 11)

- **Similitud semántica**: En los casos en los que las reglas anteriores no permitan determinar un sustantivo adecuado para reemplazar el adjetivo sustantivado, se recurre a una estrategia de aproximación semántica basada en vectores de palabras. (Véase Figura 12)

  Esta técnica parte de la representación distribuida del lenguaje: en los modelos lingüísticos como SpaCy, cada palabra (o más concretamente, cada token) está asociada a un vector numérico en un espacio multidimensional. Estos vectores codifican información semántica, de forma que palabras con significados similares están más cerca entre sí en este espacio vectorial.

  En este sistema se utiliza una lista predefinida de categorías prototípicas (“*persona*”, “*lugar*”, “*animal*”, “*cosa*”), cada una representada por su correspondiente vector. Luego, se calcula la similitud del adjetivo sustantivado (en su forma vectorial) con cada uno de estos prototipos, utilizando la medida de coseno de similitud. Véase la Figura 9, donde u es el vector del adjetivo y v el vector del prototipo.

$$\text{sim}(u, v) = \frac{u \cdot v}{\|u\| \cdot \|v\|}$$

Figura 9: Cálculo de similitud semántica con cercanía vectorial

El resultado es un valor entre -1 y 1 que indica cuánto se parecen ambos vectores.

El prototipo con mayor similitud es elegido como categoría semántica, siempre que su puntuación supere un umbral mínimo (VECTOR_THRESHOLD), establecido como 0.5. Si no hay ninguna categoría suficientemente cercana, se opta por un valor por defecto: la categoría *"persona"*.

Una vez determinada la categoría más afín (por ejemplo, *"animal"*), se genera un sustantivo concreto adaptado al género y número del adjetivo original, gracias a un diccionario creado manualmente.

Por ejemplo:

Categoría: *"animal"*

Género: femenino → Resultado: *"animales"*

Número: plural

Sin embargo, el presente método de cercanía vectorial no ha resultado muy efectivo, ya que en todos los casos el valor de *sim* no supera el umbral mínimo establecido. Podemos observar este ejemplo:

Adjetivo: indecisos

Similitudes con cada item de proptotipos:

{'persona': 0.2560740113258362, 'lugar': 0.2112099677324295, 'animal': 0.06295598298311234, 'cosa': 0.05546356365084648}

Categoría resultante: persona

Aunque la estimación está bien hecha, el valor de la similitud no es lo suficientemente alto como para ser seleccionado como candidato para la sustitución. Por ello, en los casos en los que no se ha encontrado un candidato por el método anterior, el resultante será *"personas"* por defecto.

A continuación, se pueden consultar las partes del código mencionadas anteriormente.

```
# reglas de color
if text_low in self.config.colores:
    candidato_r = 'color' if numero == ['Sing'] else 'colores'

# reglas de nacionalidades
elif text_low in self.config.nacionalidades:
        if genero == ['Fem']:
            candidato_r = 'mujer' if numero == ['Sing'] else 'mujeres'
        else:
            candidato_r = 'hombre' if numero == ['Sing'] else 'hombres'
```

Figura 10: Tratamiento de casos de colores y nacionalidad

```
def _candidato_sustantivo(self, doc, gen, num) -> str | None:
    """
    Busca en el texto un sustantivo que concuerde en géneto y número con gen y num, respectivamente.
    Devuelve el sustantivo encontrado o None.
    """
    for i in range(len(doc) - 1):
        if (doc[i].pos_ == "NOUN" and doc[i].text != self.mask) and doc[i]:
            sust = doc[i]
            if ((sust.morph.get("Gender") == gen) and (sust.morph.get("Number") == num)) :
                return sust.text
    return None
```

Figura 11: Búsqueda de candidato por sustantivo concordante

```python
def _candidato_vectorial(self, adj, gen, num) -> str | None:
    """
    Calcula la cercanía vectorial entre el adjetivo sustantivado y las palabras definidas en PROTOTIPOS.
    Devuelve el candidato con menor distancia vectorial, o "personas/s" si no llega al umbral mínimo con ninguno.
    """
    sims = {cat: (adj.similarity(tok) if adj.has_vector else 0.0) for cat, tok in self.PROTOTIPOS.items()}

    best_cat, best_sim = max(sims.items(), key=lambda kv: kv[1])
    category = best_cat if best_sim >= self.VECTOR_THRESHOLD else "persona"

    # asignación de candidato según categoría + género/número
    if category == "persona":
        if gen == ["Masc"]:
            return "hombre"  if num == ["Sing"] else "hombres"
        elif gen == ["Fem"]:
            return "mujer"   if num == ["Sing"] else "mujeres"
        else:
            return "persona" if num == ["Sing"] else "personas"
    else:
        # para lugar/animal/cosa (no marcan género)
        plural = {"lugar" : "lugares",
                  "animal" : "animales",
                  "cosa" : "cosas"}
        return category if num == ["Sing"] else plural[category]
```

Figura 12: Búsqueda de candidato por similitud semántica

Una vez determinado el sustantivo adecuado, se sustituye la etiqueta <mask> por dicho candidato. Este procedimiento se repite para todas las máscaras presentes en la frase.

Además, como se puede observar en la Figura 15, si el sustantivo elegido es *"persona"* o *"personas"* y va precedido por un determinante o cuantificador masculino (*"el"*, *"los"*, *"un"*, *"algunos"* …), se realiza una corrección automática para cambiar el determinante a su forma femenina (*"la"*, *"las"*, *"una"*, *"algunas"* …), asegurando así la concordancia gramatical de la frase final. Para ello se hace uso de un diccionario creado manualmente, véase la Figura 14.

```python
# Diccionario de género: forma_masculina → forma_femenina
gender_dict = {
    "el": "la", "la": "el", "los": "las", "las": "los",
    "un": "una", "una": "un", "unos": "unas", "unas": "unos",
    # cuantificadores
    "algún": "alguna", "algunos": "algunas",
    "todo": "toda", "todos": "todas",
    "ninguno": "ninguna", "ningunos": "ningunas"}
```

Figura 14: Diccionario de género para la transformación automática

```
# si el candidato por reglas es persona(s), se fuerza a femenino el determinante
if candidato_r in ("persona", "personas"):
    if(det_genero == ["Masc"]):
        fem = gender_dict.get(det.text())
        if det.i == 0:
            fem = fem.capitalize()
        if adv:
            frase_reglas = frase_reglas.replace(det.text() + adv.text() + candidato_r,
                                                fem + adv.text() + candidato_r)
        else:
            frase_reglas = frase_reglas.replace(det.text() + candidato_r, fem + candidato_r)
```

Figura 15: Corrección del género en caso de *"persona"* o *"personas"*

4.2.1.2 Adaptación con modelo RoBERTa

De forma paralela, se genera una segunda adaptación utilizando el modelo de lenguaje basado en RoBERTa. Mediante el uso de un pipeline de *fill-mask* de HuggingFace, el sistema solicita al modelo que proponga un término que complete la frase en el lugar de la etiqueta <mask>, generando una versión alternativa basada en predicción de lenguaje natural[17]. Véase la Figura 16.

```
def _frase_roberta(self, frase: str) -> str | None:
    """
    Rellena la aparición de <mask> usando el pipeline fill-mask.
    Devuelve la frase ya adaptada o None.
    """
    unmasker = hf_pipeline('fill-mask', model=self.config.mask_model)
    resultado = unmasker(frase)

    if not resultado: return None

    # Determinación del primer grupo de predicciones
    primero = resultado[0]

    if isinstance(primero, list):
        if not primero:
            return None
        pred = primero[0]
    else: pred = primero

    return pred['sequence']
```

Figura 16: Sustitución de <mask> mediante modelo RoBERTa

[17] La elaboración de esta parte del código se ha inspirado en un trabajo referenciado como [18] en la bibliografía.

Este enfoque aprovecha el conocimiento estadístico y contextual del modelo para ofrecer alternativas que, si bien pueden no ser perfectas en todos los casos, enriquecen la propuesta adaptativa del sistema con una perspectiva más flexible y creativa.

4.2.2 Caso 2: Adaptación de 'lo + adj'

Además de los adjetivos sustantivados introducidos por determinantes o cuantificadores (tratados mediante la etiqueta <mask>), el sistema también contempla una de las construcciones más representativas y frecuentes de sustantivación en español: la estructura "lo + adjetivo". Expresiones como *"lo importante"*, *"lo necesario"*, o *"lo desconocido"* son construcciones abstractas que pueden presentar dificultades de comprensión para personas con baja competencia lectora.

Para adaptar este tipo de estructuras, se ha optado por una estrategia de reformulación generativa basada en lenguaje natural, utilizando un modelo de lenguaje preentrenado capaz de comprender y reescribir frases bajo instrucciones explícitas. En concreto, se ha utilizado el modelo Salamandra 7B Instruct.

El método encargado de esta adaptación se denomina *process_file_lo*, dentro de la clase *AdaptationPipeline* y sigue los siguientes pasos:

Se inicializa el modelo de lenguaje (AutoModelForCausalLM) y su tokenizador asociado (AutoTokenizer) a partir del identificador definido en la configuración (self.config.lo_model). Se utiliza device_map="auto" para asignar automáticamente el modelo a la GPU disponible, y torch_dtype=torch.bfloat16 para optimizar el uso de memoria. Véase la Figura 17.

```
# Preparación del tokenizador y del modelo
tokenizer = AutoTokenizer.from_pretrained(self.config.lo_model)
model = AutoModelForCausalLM.from_pretrained(
    self.config.lo_model,
    device_map="auto",
    torch_dtype=torch.bfloat16
)
```

Figura 17: Carga del modelo Salamandra

Para cada frase del archivo *adaptar_lo*.csv, se genera un *prompt instructivo* en formato conversacional con la siguiente instrucción:

*"Por favor, adapta esta frase eliminando **todas** las instancias del pronombre «lo» sin cambiar el significado:"*

A esta instrucción se le añade la frase original, formando un mensaje estructurado que se aplica mediante el método *apply_chat_template*, que formatea el mensaje como una conversación entre usuario y asistente, siguiendo la plantilla de chat del modelo Salamandra. La entrada es codificada con el tokenizador, y se genera una respuesta con un máximo de 50 tokens nuevos para evitar resultados excesivamente largos. Véase la Figura 18.

```
for row in reader:
    frase = row[0].strip()

    message = [{"role": "user", "content": (
        "Por favor, adapta esta frase eliminando **todas** las "
        "instancias del pronombre «lo» sin cambiar el significado:\n\n"
        f"\"{frase}\""
    )}]

    date_string = datetime.today().strftime('%Y-%m-%d')

    prompt = self.tokenizer.apply_chat_template(
        message,
        tokenize=False,
        add_generation_prompt=True,
        date_string=date_string
    )

    inputs = self.tokenizer.encode(prompt, add_special_tokens=False, return_tensors="pt")
    outputs = self.model.generate(input_ids=inputs.to(self.model.device), max_new_tokens=50)
```

Figura 18: Generación de respuesta mediante *prompt* redactado

Una vez generada la salida, se decodifica la secuencia de texto y se extrae la última línea, que contiene la reformulación final propuesta por el modelo. Se eliminan comillas o símbolos tipográficos innecesarios, y se deja preparada como versión adaptada de la frase original. Finalmente, se guarda en el nuevo archivo CSV una fila con la frase original y su adaptación. Véase Figura 19.

```
    # Decodificación y post-procesamiento
    decodes = tokenizer.decode(outputs[0], skip_special_tokens=True).strip()

    # Selección de la última frase de la respuesta
    adaptada = decodes.splitlines()[-1].strip().strip('"“”')

    writer.writerow([frase, adaptada])
```

Figura 19: Decodificación de la respuesta y escritura en csv

A diferencia de los casos tratados con la etiqueta <mask>, donde el sistema genera dos versiones alternativas de adaptación (una por reglas lingüísticas y otra mediante el modelo RoBERTa), en el caso de las estructuras con *'lo'* + adjetivo se opta por generar una única adaptación mediante el modelo Salamandra.

Esta decisión se fundamenta principalmente en la limitación práctica del modelo: Salamandra, al ser un modelo de generación autoregresiva, produce una sola respuesta por defecto. Aunque sería técnicamente posible generar múltiples variaciones, esto implicaría procesos adicionales de reformulación del *prompt* o uso de técnicas de *sampling*. Además, a diferencia de los casos de <mask> donde la sustitución puede guiarse por género, número y contexto morfológico, las construcciones con *'lo'* carecen de información explícita que permita una sustitución basada en reglas fijas.

Estas expresiones son más abstractas y requieren una reformulación semántica completa más allá de simples transformaciones morfosintácticas, lo que hace que la solución por reglas resulte impracticable o ineficaz.

Por tanto, se considera que una única propuesta generada con calidad suficiente por el modelo es adecuada para cumplir con el objetivo del sistema, que es facilitar una versión más explícita y accesible del contenido original.

4.3 Elección de modelos y justificación

Uno de los componentes fundamentales del sistema desarrollado para la adaptación de adjetivos sustantivados ha sido la elección de modelos de lenguaje adecuados a las tareas específicas del proyecto. Para ello, se han seleccionado dos modelos distintos, optimizados para contextos diferentes, pero ambos pertenecientes al conjunto de recursos lingüísticos **ALIA** [19], desarrollado por el Barcelona Supercomputing Center en el marco del Plan Nacional de Tecnologías del Lenguaje y la Estrategia Nacional de Inteligencia Artificial (ENIA).

A continuación, se describe cada uno de los modelos utilizados, la tarea que desempeñan en el sistema, y la justificación de su elección.

4.3.1 Modelo para la adaptación de frases con <mask>

Para los casos en que los adjetivos sustantivados han sido detectados y marcados mediante la etiqueta <mask>, se ha empleado el modelo **RoBERTa-base-bne**, alojado en HuggingFace bajo el identificador: PlanTL-GOB-ES/roberta-base-bne.

Este modelo está basado en la arquitectura RoBERTa (una versión optimizada de BERT) y ha sido entrenado específicamente en lengua española, utilizando una gran cantidad de datos provenientes de corpus oficiales y variados (Wikipedia, prensa, textos administrativos, etc.).

La elección de este modelo responde a las siguientes razones:

- **Alta precisión en tareas de *fill-mask*:** RoBERTa ha demostrado un rendimiento superior en tareas de predicción de palabras ausentes respecto a BERT, gracias a su preentrenamiento más robusto y sin segmentación de frases.

- **Contextualización profunda**: su arquitectura es bidireccional, esto es que permite analizar el contexto anterior y posterior a la máscara, lo que resulta crucial para predecir un sustantivo adecuado que sustituya al adjetivo sustantivado. Esto es clave en textos reales, donde el significado de un adjetivo sustantivado depende del entorno gramatical y semántico.

- **Modelo entrenado en español**: al tratarse de un modelo ajustado al español de forma nativa, evita errores de traducción o pérdida de significado que podrían surgir con modelos multilingües.

- **Facilidad de integración**: su disponibilidad en HuggingFace y compatibilidad con el pipeline fill-mask ha facilitado una integración rápida y funcional en el sistema, sin necesidad de ajuste o entrenamiento adicional.

4.3.2 Modelo para la adaptación de frases con *'lo'* + adjetivo

Los casos con estructuras del tipo *'lo'* + adjetivo presentan un mayor grado de abstracción y no se pueden resolver mediante una simple sustitución, como ocurre con <mask>. Para abordarlos, se optó por una estrategia de generación de lenguaje natural, basada en *prompts* cuidadosamente diseñados y ejecutados mediante un modelo de tipo instruction-tuned.

Inicialmente se utilizó el modelo: BSC-LT/salamandra-2b-instruct, sin embargo, durante las pruebas, se observó que los resultados generados no respondían a los fines del proyecto, ya que no devolvían una respuesta satisfactoria. Por tanto, se decidió sustituirlo por su versión más potente: BSC-LT/salamandra-7b-instruct

Este segundo modelo, de mayor tamaño y capacidad, produjo mejoras significativas en la calidad de las respuestas generadas, generando definiciones más coherentes, precisas y adaptadas al objetivo de lectura fácil.

Las razones del uso de este modelo son las siguientes:

- **Diseño para tareas de instrucción (instruct)**: Salamandra ha sido ajustado específicamente para entender y responder a indicaciones escritas en lenguaje natural. Esto lo hace ideal para prompts como *"Adapta la frase..."*.

- **Entrenado en español**: al igual que RoBERTa-bne, Salamandra ha sido entrenado íntegramente en lengua española, lo cual garantiza una correcta interpretación semántica y sintáctica de los prompts.

- **Mejoras con el modelo 7B**: aunque su ejecución es más lenta debido a su tamaño, el modelo de 7 mil millones de parámetros ofreció mejoras notables en fluidez, detalle y pertinencia de las respuestas, justificando así el coste computacional adicional.

- **Alineado con los principios de accesibilidad**: la capacidad de generar explicaciones o definiciones explícitas desde un concepto abstracto (como *"lo complejo"*) es una herramienta poderosa para adaptar textos con alto nivel de inferencia a un formato más comprensible.

Como ya se ha mencionado, ambos modelos forman parte del ecosistema ALIA, lo que garantiza no solo calidad técnica y entrenamiento específico en español, sino también coherencia con los objetivos estratégicos del desarrollo de tecnologías del lenguaje en España. Su integración ha permitido combinar enfoques estadísticos avanzados con lógica lingüística estructurada, ofreciendo un sistema híbrido más robusto y adaptable.

4.4 Pruebas y validación

Durante el desarrollo del sistema, se ha seguido un enfoque iterativo de prueba y validación basado en la ejecución constante del código sobre un conjunto de frases reales. El objetivo de estas pruebas ha sido garantizar que los distintos componentes del sistema —identifiación, marcado, adaptación por reglas y por modelos— funcionaran correctamente y ofrecieran resultados coherentes y útiles desde el punto de vista de la accesibilidad lingüística.

4.4.1 Conjunto de datos utilizado

Desde las primeras fases del proyecto se utilizó un archivo base, denominado *textos.csv*, como corpus de evaluación y prueba. Este archivo contiene más de 300 frases[18], seleccionadas por su relevancia para el objetivo del sistema. El conjunto incluye tanto frases con adjetivos sustantivados como frases sin casos problemáticos, permitiendo así evaluar no solo la capacidad de detección, sino también la robustez del sistema ante textos neutros.

A continuación, en la Figura 19, se muestra un fragmento del archivo de frases de prueba, indicando si cada oración presenta o no problemática. Esta clasificación se validó de manera manual por un lingüista: Isam Diab, el cotutor del presente TFG.

[18] Las frases que forman el corpus de prueba han sido extraídas de diferentes fuentes. Algunas pertenecen al corpus de la Real Academia Española (RAE): https://www.rae.es/.
Otras pertenecen a fuentes no formales (Véase en las referencias [20],[21],[22],[23] de la bibliografía del presente trabajo).

Frases de prueba	Presenta la problemática
La mudanza de Tomás	No
Nueva derrota de los Yankees	No
El perro, el amigo del hombre	No
El gato, el amigo de las brujas	No
De todos los colores, el rojo es el que más me gusta.	Sí
Lo más bonito de Barcelona es la Sagrada Familia.	Sí
Los italianos son muy atractivos.	Sí
El más alto seguramente sea el mejor jugador de baloncesto.	Sí
Esta respuesta es incorrecta, la correcta es la segunda opción.	Sí
Lo bueno es que aún tienes otra oportunidad.	Sí
Yo no cogería la camisa roja, la azul es más bonita.	Sí
No seas superficial, las rubias también pueden ser inteligentes.	Sí
El entrenador puso a los más rápidos para salir al contraataque.	Sí
Lo mejor aún está por venir.	Sí
A Andrea le gustan los morenos.	Sí
¿Qué vestido te gusta más? – El verde es el más bonito.	Sí
Los ingleses son muy pálidos	Sí

Figura 19: Muestra del corpus de frases con y sin problemática.

La validación del sistema no se ha basado en una única ejecución final, si no que se ha realizado de forma paralela al desarrollo del proyecto. Cada vez que se introducía una modificación en el código —por ejemplo, una nueva regla de identificación, una corrección morfosintáctica o un ajuste en la generación de sustituciones— se ejecutaba de nuevo el procesamiento sobre el archivo textos.csv.

Este procedimiento ha permitido comprobar de inmediato los efectos del cambio, observando si las frases afectadas eran procesadas correctamente. Además, ha facilitado la detección de errores no previstos, lo que ha contribuido a mejorar la estabilidad general del sistema.

4.4.2 Pruebas dirigidas con frases problemáticas

Además del corpus general, se ha adoptado una estrategia complementaria basada en archivos de prueba específicos para casos problemáticos. En varias ocasiones, al detectar que una frase concreta no era tratada correctamente (por ejemplo, por un fallo en el etiquetado gramatical, una mala interpretación del contexto o una sustitución inadecuada), se extraía esa frase y se incluía en un archivo de pruebas reducido, llamado *prueba.csv*, dedicado exclusivamente a evaluar ese caso.

Este método permitió realizar pruebas más controladas y precisas, aislando los efectos de una modificación del código sobre un único caso. Esto resultó especialmente útil durante la implementación del componente *sustantivados_component* y del sistema de adaptación por reglas, donde ajustes muy específicos podían corregir un fallo sin afectar al comportamiento general.

Algunos de los fallos que se identificaron y corrigieron gracias a esta estrategia fueron:

- **Inserción incompleta de máscaras**: En las primeras versiones del componente de detección, el sistema solo insertaba una única etiqueta <mask> por frase, incluso si había múltiples adjetivos sustantivados presentes. Al probar individualmente frases con varios casos, se detectó este fallo y se modificó la lógica para permitir múltiples inserciones en una misma oración.

  Un ejemplo es la frase: *"Los deshonestos dominan el mundo, pero los honestos duermen mejor."*, en la que el proceso de inserción de <mask> debe realizarse en dos puntos (delante de *"deshonestos"* y de *"honestos"*).

- **Sustitución incompleta durante la adaptación**: Algo similar ocurría en la etapa de adaptación: solo se reemplazaba la primera aparición de <mask>, dejando el resto sin procesar. Las pruebas aisladas sobre frases con múltiples máscaras permitieron comprobar este error y adaptar el sistema para procesar todas las ocurrencias de forma iterativa y coherente. La solución a este problema consistió en la búsqueda de candidatos por cada aparición de la máscara, en vez de por cada frase. Como se ha podido ver en la Figura 8, el código de adaptación se aplica de forma iterada sobre la lista de las posiciones de cada máscara dentro de la fila leída.

- **Errores al recuperar el adjetivo sustantivado o el determinante**: En algunas frases, el sistema fallaba al identificar correctamente el adjetivo que debía ser sustituido o el determinante que le daba género y número. Al probar estas frases de forma individual, fue posible trazar con precisión el recorrido del análisis y ajustar los índices o condiciones para mejorar la detección y extracción de estos elementos.

- **Limitaciones del etiquetado de spaCy**: Las pruebas individuales también ayudaron a discernir cuándo un fallo no era imputable al código del sistema, si no a errores en el etiquetado morfosintáctico realizado por el modelo de spaCy. Por ejemplo, en la *frase "La pelirroja está leyendo un fragmento de Cervantes."*, el adjetivo *"pelirroja"* es etiquetado incorrectamente como sin género, lo que dificulta la concordancia gramatical durante la adaptación.

  En otros casos, ciertos adjetivos sustantivados eran clasificados como sustantivos y viceversa. Algunos de estos casos fueron solventados como se ha mostrado en el apartado 4.1.1 de esta Memoria.

Durante el desarrollo de la adaptación del caso de *'lo'* + adjetivo, uno de los principales retos consistió en diseñar un *prompt* eficaz que guiara al modelo para generar una explicación clara, coherente y orientada a lectura fácil. En las primeras pruebas, los resultados eran vagos: no eliminaban la partícula *'lo'* de la oración, cambiaban el significado de la frase o, incluso, devolvían la misma frase de entrada sin ninguna modificación.

Para resolver esto, se realizaron múltiples iteraciones con distintos formatos de *prompt*, evaluando cómo respondía el modelo a pequeñas variaciones. Se probaron diversas estrategias, como la creación de instrucciones más explícitas (*"NO puede aparecer el pronombre 'lo' en la siguiente frase…"*), o la repetición de requisitos (*"Adapta esta frase eliminando las apariciones de 'lo' de la frase. No tiene que haber ningún 'lo' en la adaptación…"*).

Finalmente, se optó por una fórmula directa y comprensible, centrada en la instrucción *"Por favor, adapta esta frase eliminando **todas** las instancias del pronombre «lo» sin cambiar el significado:"*, que proporcionó respuestas más precisas, coherentes y adaptables a los fines del proyecto.

4.4.3 Validación

Tras haber completado y ajustado el código según las observaciones tomadas durante el desarrollo, se ha realizado una evaluación cuantitativa de los resultados. Los criterios de validación seguidos han sido los siguientes

- **Claridad**: que la frase sea fácilmente comprensible por una persona con dificultades lectoras.
- **Relevancia semántica**: que la frase se mantenga dentro del campo de significado de la oración original.
- **Corrección gramatical y lexical**: que la adaptación sea lingüísticamente válida y coherente con la estructura de la frase original.

El sistema ha sido evaluado mediante un corpus compuesto por 322 frases con distintos niveles de complejidad, incluyendo tanto construcciones problemáticas como ejemplos sin dificultad, para comprobar la precisión y robustez del sistema.

Los resultados se analizan desde una doble perspectiva:

Por un lado, la identificación de adjetivos sustantivados y su correcta marcación con la etiqueta <mask> o la identificación de estructuras del tipo *'lo'* + adjetivo.

Por otro, la adaptación automática de estas construcciones, tanto mediante reglas lingüísticas como mediante modelos de lenguaje preentrenados.

A continuación, se detallan los resultados obtenidos en ambas etapas, junto con ejemplos representativos y un análisis de su adecuación desde el punto de vista de la accesibilidad lingüística.

4.5 Resultados Fase de Identificación

Tras el desarrollo completo del sistema, se procedió a una fase de evaluación cuantitativa que permitiera comprobar de forma objetiva su eficacia en la detección de adjetivos sustantivados. A diferencia de las pruebas realizadas durante la etapa de desarrollo —más centradas en ajustes funcionales y detección de errores puntuales—, esta fase tiene como objetivo valorar el rendimiento del sistema ya consolidado, desde un punto de vista medible.

Esta evaluación se llevó a cabo sobre el conjunto ya mencionado de frases, las cuales sirvieron como un grupo de referencia (*gold standard*) sobre el que comparar el rendimiento del sistema. Como ya se ha mencionado previamente, entre estas frases, había tanto construcciones problemáticas, como frases sin dichas estructuras, lo que permitió evaluar la capacidad del sistema tanto para detectar correctamente como para no generar falsos positivos.

De entre el conjunto de prueba, 139 frases fueron etiquetadas como "problemáticas", es decir, contenían al menos un adjetivo sustantivado, mientras que las 183 frases restantes no presentaban este tipo de dificultad.

Una vez ejecutado el sistema sobre este conjunto, se obtuvo un resultado global satisfactorio: 279 de las 322 frases fueron clasificadas correctamente, lo que representa un 86,65% de acierto general. Este porcentaje incluye tanto los casos positivos correctamente detectados como aquellos negativos en los que el sistema acertó al no marcar ninguna problemática.

El balance general de detecciones puede visualizarse en el gráfico circular de la Figura 20, donde se muestra de forma clara que la mayoría de las frases fueron procesadas con éxito.

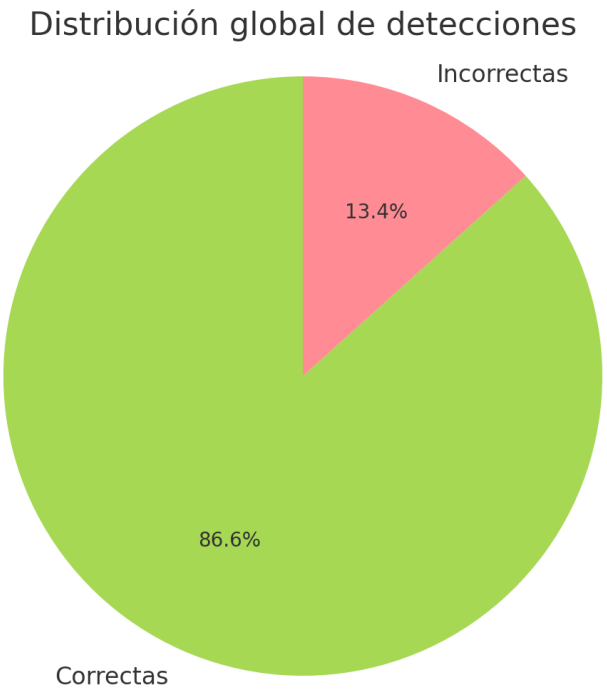

Figura 20: Distribución global de frases clasificadas correcta e incorrectamente por el sistema.

Al analizar con mayor detalle los resultados, se observó que dentro del grupo de frases con adjetivos sustantivados (139 en total), el sistema logró detectar correctamente 89 de ellas, lo que representa un 65,44% de acierto específico en los casos con problemática. Aunque este porcentaje indica que hay margen de mejora, sobre todo en estructuras más complejas o ambiguas, también confirma que el sistema cumple razonablemente bien su objetivo principal: identificar construcciones que pueden dificultar la comprensión lectora.

Una diferencia significativa se observó entre los dos tipos de estructuras contempladas en el sistema. En los casos con estructuras del tipo *'lo'* + adjetivo, el rendimiento fue muy alto: el sistema detectó correctamente 20 de las 21 frases de este tipo, lo que representa una tasa de acierto del 95,24%. Esta alta precisión puede atribuirse a la simplicidad del patrón de detección, y a la poca variación en los casos que cubre.

En cambio, los resultados fueron algo más modestos en los casos restantes (det. + adj. / cuan. + adj.…). De las 114 frases etiquetadas manualmente como pertenecientes a esta categoría, el sistema acertó en 78 casos, alcanzando así un 68,42% de detección correcta. Se puede ver que este tipo de construcción es más difícil de identificar de forma automática, probablemente debido a ambigüedades en el análisis morfológico y a ciertas limitaciones del etiquetado POS automático.

Este desglose puede observarse en el siguiente gráfico de barras de la Figura 21.

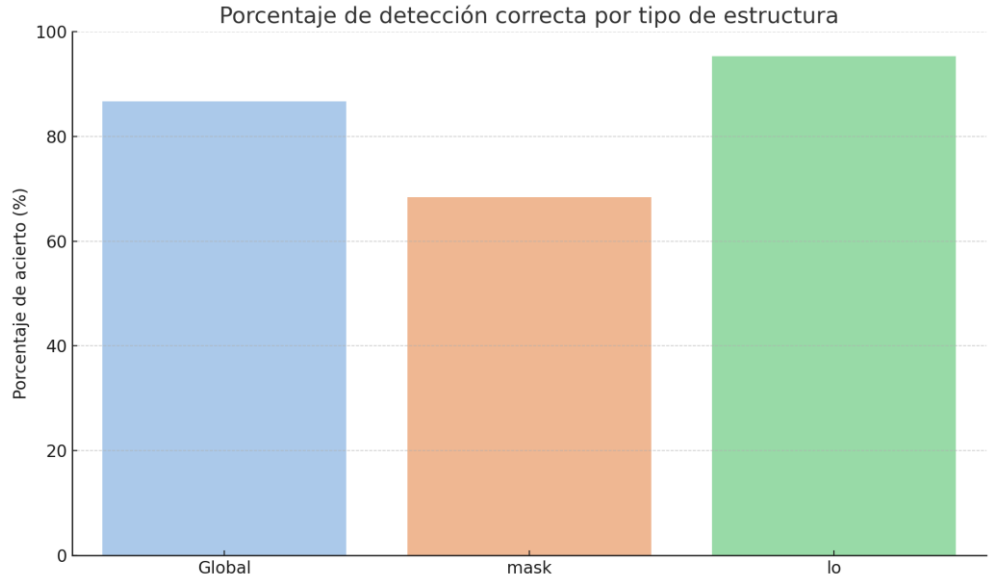

Figura 21: Porcentaje de detección correcta global y por tipo de estructura ("mask" y "lo").

Para complementar este análisis, también se ha elaborado una matriz de confusión para visualizar los aciertos y errores según el cruce entre los datos reales (anotaciones manuales) y la predicción del sistema. Esta matriz refleja que, de las 139 frases con problemática real, el sistema detectó correctamente 100 y no detectó 39. Por otro lado, de las 183 frases sin problemática, solo 4 fueron erróneamente clasificadas como problemáticas, mientras que las otras 179 fueron correctamente ignoradas. Estos datos demuestran no solo una buena capacidad de detección, sino también una baja tasa de falsos positivos, lo cual es especialmente valioso en un sistema que debe ser integrado como herramienta de apoyo para la adaptación de textos.

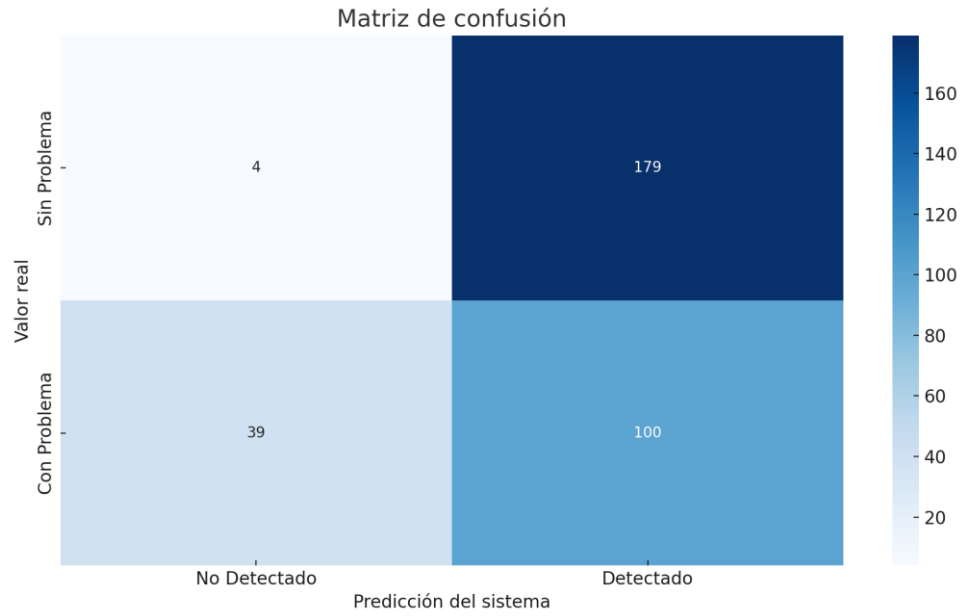

Figura 22: Matriz de confusión generada a partir del conjunto de validación manual.

En conjunto, los resultados de esta evaluación final confirman que el sistema es fiable y preciso, especialmente en su objetivo de identificar adjetivos sustantivados problemáticos para la lectura fácil. Aunque el rendimiento en frases del tipo mask aún presenta un considerable margen de mejora, la altísima precisión en estructuras con *'lo'* demuestra la robustez del enfoque basado en patrones lingüísticos y reglas personalizadas.

Tras la vista de estos resultados, se pretende investigar las causas de los casos mal detectados, en especial los de inserción de <mask>. Después de una vista general de las frases que fallaron, así como la realización de comprobaciones observando las características de las palabras, se detecta que un gran porcentaje de los errores deriva del etiquetado realizado por SpaCy. Como se ha explicado en la sección de pruebas de la presente Memoria, fue necesario el reetiquetado de ciertos tokens (los colores y nacionalidades) antes de procesar los patrones. No obstante, se han encontrado numerables casos, aparte de los mencionados, en los que la componente "POS" de los tokens correspondientes a adjetivos sustantivados contiene la característica "NOUN", en vez de "ADJ". Estos casos no son palabras generales como los colores y nacionalidades, por lo que no es factible un pre-etiquetado de cada uno de los adjetivos posibles. Esto ha imposibilitado la correcta identificación de los patrones definidos para la detección.

Como posible solución, se intentó identificar el adjetivo sustantivado por otros medios que no fueran accediendo a su etiqueta de categoría gramatical. SpaCy cuenta con otras etiquetas, como "TAG" o "DEGREE", que permiten distinguir a un adjetivo del resto de palabras, aunque no se haya etiquetado como tal. Sin embargo, estos servicios de SpaCy no se encuentran integrados en el modelo usado para la adaptación, así como tampoco cuentan con ellos el resto de los modelos en español.

Otra causa de los falsos negativos generados es la presencia de palabras polisémicas, es decir que tienen varios significaos y, por tanto, pueden etiquetarse con más de una categoría gramatical. Este es el caso de la palabra *"café"* encontrada en la frase de prueba: *"El café de sus ojos es lo que más me gusta."*, en la cual actúa de adjetivo describiendo un color, pero debido a su doble significado, SpaCy lo reconoce como un sustantivo.

Aunque son muchos menos casos, esto ocurre de forma similar con los falsos positivos, se detectan palabras que pueden ser adjetivos o sustantivos según el contexto. En una de las frases del corpus: *"Mi flaca está cansada, ya es hora de retirarnos."*, la palabra *"flaca"* no se usa con su sentido literal (persona delgada), si no como un término coloquial y afectivo, muy común en algunas regiones hispanohablantes. En este contexto, *"mi flaca"* equivale a *"mi novia"* o *"mi chica"*, por lo que es considerado un sustantivo, sin embargo, el sistema aplica su etiqueta más común, ADJ, por lo que clasifica la frase como problemática sin serlo.

4.6 Resultados Fase de Adaptación

Siguiendo la línea del análisis anterior, se ha procedido a la observación de los resultados de la fase encargada de adaptar las frases previamente clasificadas.

El análisis de resultados se ha dividido en dos bloques correspondientes a los dos tipos de construcciones detectadas: las frases marcadas con <mask> y las frases con estructuras del tipo *'lo'* + adjetivo. En conjunto, considerando ambos tipos de estructuras, el sistema ha logrado adaptar correctamente aproximadamente el 79% de las frases evaluadas.

Se ha combinado una evaluación cuantitativa de la eficacia de cada método con una valoración cualitativa de la calidad de las reformulaciones generadas.

4.6.1 Resultados adaptación de <mask>

Una vez detectadas las frases que contienen adjetivos sustantivados marcadas con la etiqueta <mask>, el sistema genera dos versiones adaptadas: una utilizando un conjunto de reglas lingüísticas y otra mediante el modelo de lenguaje preentrenado RoBERTa. Este enfoque doble tiene como objetivo aprovechar las ventajas de ambas estrategias: la precisión morfosintáctica de las reglas y la capacidad contextual del modelo neuronal.

Se evaluó un total de 77 frases, y tras una revisión manual se determinó que 63 fueron correctamente adaptadas por al menos uno de los dos métodos, lo que representa una cobertura general del sistema del 81,82%.

Desglosando los resultados por método:

- Las reglas ofrecieron una adaptación correcta en 55 frases, con una precisión del 71,43%.
- RoBERTa fue capaz de generar una adaptación válida en 42 frases, con una precisión del 54,55%.

Es importante destacar que 21 frases fueron resueltas únicamente por las reglas, mientras que 8 lo fueron solo por RoBERTa, y 34 frases fueron correctamente adaptadas por ambos métodos. Esto refuerza la utilidad del enfoque híbrido, ya que permite compensar las debilidades de cada técnica individual.

La Figura 23 representa gráficamente esta distribución de aciertos.

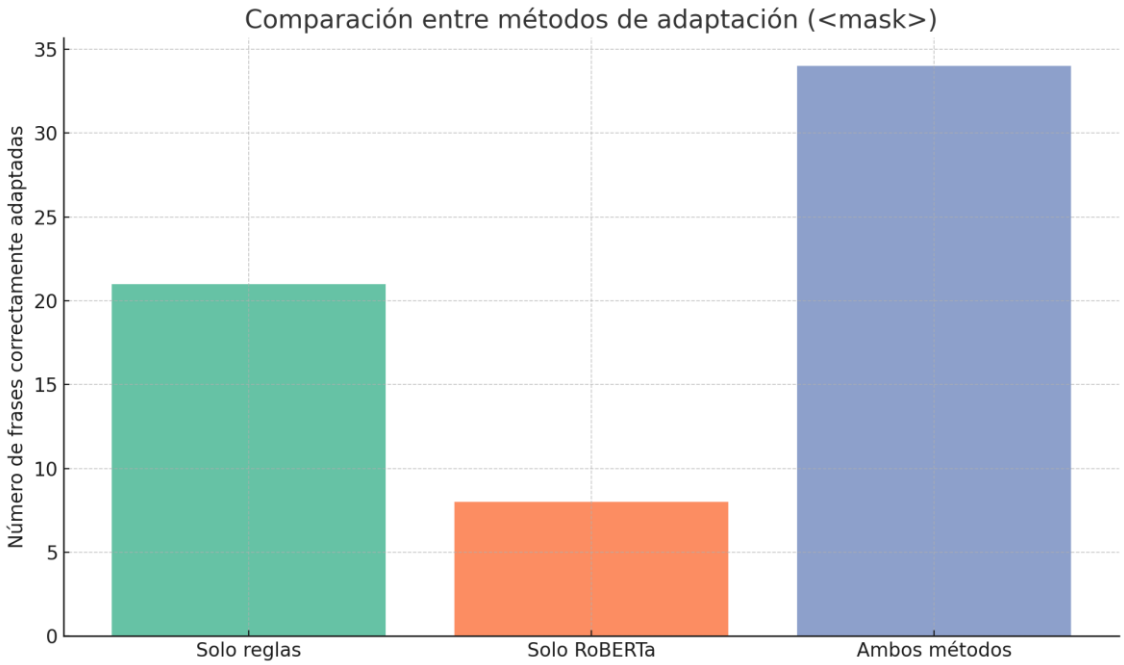

Figura 23: Número de frases correctamente adaptadas por cada método.

El porcentaje relativamente bajo de eficacia en la adaptación de frases mediante RoBERTa puede explicarse por la propia naturaleza del modelo y su objetivo de entrenamiento. El modelo está diseñado para completar huecos (*fill-mask*) generando palabras que encajen gramatical y semánticamente en el contexto, pero no está específicamente orientado a resolver el fenómeno de los adjetivos sustantivados. En muchos casos, RoBERTa sustituye la máscara por adverbios, adjetivos o incluso determinantes, generando frases bien formadas y coherentes, pero que no eliminan ni reformulan la sustantivación, por lo que no resuelven el problema de accesibilidad planteado según la MLF. Esto evidencia la necesidad de combinar el modelo con criterios más controlados, como los que aportan las reglas gramaticales.

Además de la evaluación cuantitativa, se seleccionaron ejemplos representativos para comparar cómo se comportan ambos métodos ante diferentes tipos de frase. En algunos casos, las reglas producen versiones más neutras y sistemáticas, mientras que RoBERTa genera adaptaciones más contextualizadas, aunque a veces menos controladas. La siguiente tabla (Figura 24) recoge varios casos ilustrativos.

	Frase original	Adaptación por reglas	Adaptación por RoBERTa	Tipo de acierto
1	Los <mask> italianos son muy atractivos.	Los hombres italianos son muy atractivos.	Los vinos italianos son muy atractivos.	Solo reglas
2	Los <mask> ingleses son muy pálidos	Los hombres ingleses son muy pálidos	Los jugadores ingleses son muy pálidos	Solo reglas
3	Primero María, la <mask> de Madrid	Primero María, la mujer de Madrid	Primero María, la madre de Madrid	Solo reglas
4	Yo no cogería la camisa roja, la <mask> azul es más bonita.	Yo no cogería la camisa roja, la color azul es más bonita.	Yo no cogería la camisa roja, la camisa azul es más bonita.	Solo RoBERTa
5	Uso dos autos: el <mask> viejo para ir al trabajo y el <mask> nuevo para ocasiones	Uso dos autos: el trabajo viejo para ir al trabajo y el trabajo nuevo para ocasiones especiales.	Uso dos autos: el auto viejo para ir al trabajo y el auto nuevo para ocasiones especiales.	Solo RoBERTa
6	La policía de Boston es menos estricta que la <mask> de Nueva York.	La policía de Boston es menos estricta que la mujer de Nueva York.	La policía de Boston es menos estricta que la policía de Nueva York.	Solo RoBERTa
7	De todos los colores, el <mask> rojo es el que más me gusta.	De todos los colores, el color rojo es el que más me gusta.	De todos los colores, el color rojo es el que más me gusta.	Ambos
8	Esta respuesta es incorrecta, la <mask> correcta es la segunda opción.	Esta respuesta es incorrecta, la respuesta correcta es la segunda opción.	Esta respuesta es incorrecta, la respuesta correcta es la segunda opción.	Ambos
9	El <mask> azul es triste	El color azul es triste	El color azul es triste	Ambos

Figura 24: Ejemplos comparativos de adaptación por reglas y por RoBERTa.

Este análisis permite identificar patrones de rendimiento: por ejemplo, las reglas tienden a funcionar mejor en casos donde hay información morfológica clara, como en estructuras con determinantes o nacionalidades, mientras que RoBERTa acierta con más facilidad cuando el contexto semántico es rico, aunque puede desviarse del significado original.

Los resultados obtenidos muestran que el sistema de adaptación para <mask> logra un buen nivel de cobertura y precisión. La combinación de reglas lingüísticas y un modelo de lenguaje como RoBERTa permite abordar distintos tipos de frase con eficacia, reforzando la idea de que un enfoque híbrido es más robusto y adaptable que una estrategia única.

En la práctica, esto permite que, en un contexto real de producción de textos accesibles, el sistema ofrezca una doble propuesta de adaptación, dando margen al revisor o adaptador humano para elegir la versión más adecuada según el público objetivo o el contenido específico del texto.

4.6.2 Resultados adaptación 'lo' + adj

Para las frases con estructuras del tipo *'lo'* + adjetivo, el sistema genera una única versión adaptada utilizando el modelo generativo Salamandra-7B-Instruct. Estas construcciones, que suelen expresar conceptos abstractos (*"lo importante"*, *"lo desconocido"*, etc.), presentan un reto particular en términos de adaptación, ya que requieren no solo una transformación gramatical, sino una reformulación semántica del contenido.

Se evaluó un total de 22 frases con construcciones de este tipo. Tras la validación manual de las adaptaciones generadas, 15 frases fueron consideradas correctamente adaptadas, lo que representa un 68,18% de acierto.[19] 7 frases fueron clasificadas como incorrectas o dudosas, es decir, no cumplían completamente con el objetivo de mantener el sentido original o presentaban ambigüedades, errores gramaticales o reformulaciones poco naturales.

La Figura 25 resume esta distribución.

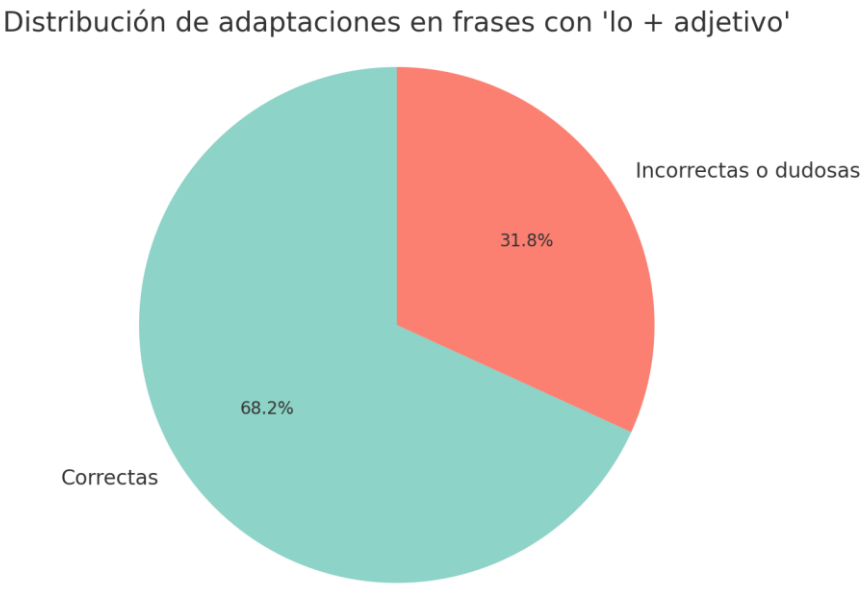

Figura 25: Distribución de adaptaciones en frases con "lo + adjetivo".

En general, el modelo ha sido capaz de generar reformulaciones comprensibles y adecuadas en una mayoría de casos. Por ejemplo:

Frase original: *"Lo desconocido da miedo."*

Adaptación generada: *"Lo que no conocemos da miedo."*

[19] Se pueden consultar los resultados de la adaptación en el Anexo 2.

La adaptación transforma una abstracción en una expresión concreta, sin perder el significado.

Entre los aciertos más destacados se encuentran frases donde el modelo fue capaz de sustituir *'lo'* + adjetivo por estructuras como *"lo que es..."*, *"la cosa que es..."* o reformulaciones con sustantivos explícitos que clarifican el mensaje.

Sin embargo, también se observaron algunas limitaciones; en ciertos casos, el modelo repite la palabra original sin realizar una transformación útil. Otras veces, el resultado incluye frases que suenan forzadas o excesivamente generales. La precisión depende mucho de la calidad del *prompt* y de la claridad de la frase original.

El módulo de adaptación para estas frases demuestra una eficacia razonable, con un acierto superior al 68%. Dada la dificultad de reformular construcciones abstractas, este resultado es positivo, especialmente teniendo en cuenta que se trata de una tarea compleja incluso para humanos. El uso de modelos instructivos ofrece una vía prometedora para abordar este tipo de adaptación de forma flexible y contextualizada.

5 Conclusiones y líneas futuras

Tras el desarrollo del sistema, y una vez alcanzadas las fases finales de implementación y evaluación, se va a realizar una reflexión sobre los resultados obtenidos, las limitaciones encontradas durante el proceso, y las posibles mejoras o extensiones futuras del proyecto. Este capítulo recoge, por tanto, las principales conclusiones derivadas del trabajo realizado, así como las líneas de trabajo que podrían explorarse para dar continuidad y proyección al sistema propuesto. Además, se incluye un análisis del impacto potencial del proyecto en diferentes ámbitos, con especial atención a su contribución a los Objetivos de Desarrollo Sostenible de la Agenda 2030.

5.1 Conclusiones:

Este Trabajo de Fin de Grado ha tenido como objetivo principal el diseño e implementación de un sistema capaz de identificar y adaptar automáticamente estructuras lingüísticas complejas en español, específicamente adjetivos sustantivados, con el fin de hacer los textos más accesibles siguiendo los principios de la MLF. A través de un enfoque híbrido, que combina análisis lingüístico tradicional mediante reglas gramaticales con modelos de lenguaje basados en aprendizaje profundo, se ha conseguido construir un sistema funcional, evaluado sobre un conjunto amplio de frases reales.

5.1.1 Logros alcanzados

Entre los principales logros obtenidos se encuentran:

El **desarrollo de una arquitectura modular y extensible**, capaz de distinguir entre diferentes tipos de estructuras problemáticas y tratarlas por separado con mecanismos de adaptación adecuados a cada caso.

La **integración de dos modelos de lenguaje avanzados** y entrenados en español (RoBERTa-bne y Salamandra 7B Instruct), lo que ha permitido adaptar las frases con un alto grado de coherencia lingüística y contextual.

La **implementación de un sistema de reglas** que permite sustituir adjetivos sustantivados por sustantivos explícitos, aprovechando tanto las marcas gramaticales del texto como listas semánticas y mecanismos de similitud vectorial.

La **validación del sistema** sobre un corpus real de más de 300 frases, con resultados que muestran una tasa global de acierto del 86,65% en la identificación, y una cobertura de adaptación del 79%

Estos resultados son especialmente relevantes teniendo en cuenta que los fenómenos lingüísticos tratados (sustantivación, elipsis, abstracción semántica) no son triviales, y representan desafíos incluso para adaptadores humanos.

5.1.2 Limitaciones encontradas

Durante el desarrollo del proyecto, se han identificado también una serie de limitaciones que conviene tener en cuenta:

Errores en el etiquetado gramatical (PoS) de spaCy: como ya se ha mencionado en el apartado de pruebas; en numerosos casos, adjetivos sustantivados fueron etiquetados incorrectamente como sustantivos, lo que impidió su detección por los patrones del sistema. Aunque se corrigieron casos comunes (colores, nacionalidades), el resto de errores son difíciles de prever o corregir manualmente sin incurrir en falsos positivos.

Limitaciones computacionales en el uso de modelos grandes: en la adaptación de estructuras con *'lo'* + adjetivo se utilizó el modelo Salamandra-7B, que, aunque mejoró sustancialmente la calidad de las respuestas respecto a su versión 2B, supuso tiempos de ejecución muy largos (varias horas). Esta lentitud se debe probablemente a la capacidad limitada del equipo local utilizado, por lo que se plantea como mejora el despliegue del sistema en servidores con mayor capacidad de cómputo, o bien el uso de variantes optimizadas del modelo.

Tratamiento de frases mixtas: en aquellas frases que contienen tanto estructuras <mask> como construcciones con *'lo'* + adj., el sistema actual no ofrece una solución integral, ya que cada subtipo se adapta por separado en archivos distintos. Se trata de un caso límite poco frecuente, pero que pone de manifiesto la necesidad de una fase posterior de consolidación de adaptaciones o una arquitectura más integrada.

Asimismo, es importante destacar que, si bien el sistema ofrece un alto grado de automatización y cobertura, la supervisión humana sigue siendo necesaria, especialmente en aquellos casos en los que se generan dos propuestas de adaptación alternativas, como ocurre con las frases marcadas con <mask>. La intervención de una persona con criterio lingüístico y conocimiento del público objetivo resulta esencial para seleccionar la opción más adecuada en función del contexto, el tono del texto y las necesidades específicas de comprensión. Esta validación no solo asegura la calidad de la adaptación final, sino que también evita posibles interpretaciones erróneas derivadas de ambigüedades o decisiones semánticas complejas.

5.1.3 Conclusiones personales

A nivel personal, el desarrollo de este proyecto ha supuesto un gran reto técnico, metodológico y organizativo. Desde la adquisición de conocimientos avanzados en PLN y uso de modelos de lenguaje, hasta la consolidación de buenas prácticas de programación y documentación, el trabajo ha sido una experiencia de aprendizaje integral.

Además, trabajar en una solución orientada a la accesibilidad ha sido especialmente enriquecedor. Ha reforzado mi compromiso con el desarrollo de tecnologías que no solo sean eficientes, si no también éticas, inclusivas y centradas en las personas.

5.1.4 Líneas futuras

Este proyecto abre múltiples vías de mejora y ampliación, tanto en términos técnicos como funcionales:

Integración en una aplicación web: el sistema ha sido diseñado con vistas a una futura implementación en una plataforma web. Esta integración permitiría ofrecer el servicio a usuarios reales (como adaptadores de contenido, docentes o profesionales del ámbito social), facilitando su uso mediante una interfaz accesible y práctica.

Mejora del rendimiento del sistema de identificación: podrían incorporarse técnicas de aprendizaje supervisado para complementar los patrones de spaCy, entrenando clasificadores que aprendan a reconocer construcciones problemáticas incluso cuando el etiquetado PoS es incorrecto.

Reformulación múltiple y explicativa con Salamandra: en la adaptación de *'lo'* + adj. podría explorarse la posibilidad de generar varias alternativas de reformulación, o de acompañar la adaptación con una explicación en lectura fácil que refuerce la comprensión del mensaje.

Extensión a otras estructuras problemáticas: el sistema actual se centra únicamente en adjetivos sustantivados, pero existen otros fenómenos lingüísticos problemáticos desde la perspectiva de la MLF (pasivas, perífrasis verbales, estructuras subordinadas…) que podrían incorporarse en versiones futuras del sistema.

Validación con usuarios reales: sería muy valioso realizar una evaluación cualitativa del sistema con lectores que pertenezcan al público objetivo (personas con discapacidad intelectual, mayores, migrantes…). Esto permitiría comprobar la eficacia real de las adaptaciones en un entorno práctico.

En definitiva, este proyecto constituye un primer paso hacia el desarrollo de herramientas tecnológicas accesibles que promuevan la inclusión lectora. La base construida puede servir como punto de partida para soluciones más amplias y sofisticadas, en las que el lenguaje sea un puente, y no una barrera, para la participación y la comprensión.

5.2 Análisis de impacto

El desarrollo del presente Trabajo de Fin de Grado, tiene un impacto potencial en distintos niveles: social, empresarial, económico, medioambiental y cultural. A continuación, se analizan los principales efectos que puede tener esta solución, tanto positivos como negativos, junto con su alineación con los Objetivos de Desarrollo Sostenible[20] (ODS) de la Agenda 2030.

El impacto social es posiblemente uno de los más relevantes. La lectura fácil es una metodología reconocida internacionalmente para mejorar la accesibilidad a la información, especialmente para personas con discapacidad intelectual, personas mayores, migrantes en proceso de aprendizaje del idioma, o personas con trastornos del lenguaje o del aprendizaje.

Este sistema permite automatizar parte del proceso de localización y adaptación de estructuras lingüísticas complejas, lo cual podría integrarse en herramientas de ayuda a la redacción accesible, asistentes de lectura, o sistemas educativos adaptativos. Facilitar la comprensión textual de manera automatizada puede contribuir a una mayor autonomía personal, participación ciudadana y acceso igualitario a los contenidos.

Este enfoque se alinea con el *ODS 4: Educación de calidad*, al promover materiales educativos accesibles; y con el *ODS 10: Reducción de las desigualdades*, al contribuir a eliminar barreras lingüísticas que dificultan la inclusión de ciertos colectivos.

Desde el punto de vista empresarial, este proyecto demuestra la viabilidad de aplicar técnicas de PLN y modelos de lenguaje avanzados para mejorar la accesibilidad textual. Podrían beneficiarse directamente las empresas que gestionen contenidos digitales accesibles. Además, la automatización del proceso puede suponer una diferenciación respecto a otras organizaciones comprometidas con la accesibilidad y la responsabilidad social.

El impacto económico se observa principalmente en la posibilidad de reducir los costes asociados a la adaptación manual de contenidos. Actualmente, adaptar textos a lectura fácil requiere intervención humana especializada, lo cual puede ser costoso y lento. Un sistema semiautomatizado como el desarrollado puede servir como soporte inicial que agilice este proceso, dejando la fase de revisión a profesionales, con un ahorro considerable de tiempo y recursos.

---

20 https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

Respecto al impacto medioambiental, se ha optado por el uso de modelos preentrenados, sin necesidad de reentrenamiento, lo cual minimiza el consumo computacional. Esta decisión equilibra eficacia y sostenibilidad, en línea con el *ODS 12: Producción y consumo responsables* y el *ODS 13: Acción por el clima*.

Desde una perspectiva cultural, el proyecto contribuye a la democratización del lenguaje. Muchas expresiones complejas del lenguaje escrito tradicional (como los adjetivos sustantivados) suponen una barrera para una parte importante de la población. Adaptar estas construcciones no solo mejora la comprensión, si no que acerca a los lectores a textos jurídicos, informativos o literarios que de otro modo resultarían inaccesibles.

El sistema, al estar diseñado específicamente para el español, también pone en valor los avances tecnológicos en nuestra lengua, en línea con los esfuerzos de la comunidad lingüística y científica hispanohablante por desarrollar herramientas propias y culturalmente relevantes. Esto refuerza el *ODS 16: Paz, justicia e instituciones sólidas*, al favorecer el acceso comprensible a información legal y administrativa, y el *ODS 17: Alianzas para lograr los objetivos*, al apoyarse en infraestructuras públicas como los modelos del PlanTL.

Aunque el sistema promueve la accesibilidad y responde a los ODS relacionados con la inclusión, también presenta algunos riesgos. El uso de modelos pesados como Salamandra-7B puede dificultar su adopción en contextos con recursos técnicos limitados, lo que podría generar desigualdad en el acceso a esta tecnología. Además, si las adaptaciones se aplican sin supervisión humana, podrían producirse simplificaciones incorrectas o interpretaciones erróneas que afecten negativamente a la calidad del contenido adaptado. Por ello, es fundamental que su implementación contemple mecanismos de control y validación para garantizar un uso ético y equitativo.

6 Bibliografía

[1] Plena Inclusión. (n.d.). Lectura fácil. [Online]. Available: https://www.plenainclusion.org/discapacidad-intelectual/recurso/lectura-facil/

[2] Instituto de Ingeniería del Conocimiento. (2025, Jan. 28). Lanzamos ERAS: el analizador de lectura fácil en español del IIC [Online]. Available: https://www.iic.uam.es/procesamiento-del-lenguaje-natural/lanzamos-eras-analizador-de-lectura-facil-en-espanol/

[3] ABC. (2024, Oct. 17). El CENID y la UA culminan el desarrollo de SimpleText para facilitar la lectura a personas con dificultades de comprensión [Online]. Available: https://www.abc.es/espana/comunidad-valenciana/cenid-culminan-desarrollo-simpletext-facilitar-lectura-personas-20241017155821-nt.html

[4] HULAT, Universidad Carlos III de Madrid. (n.d.). Plataforma EASIER, una ayuda en la comprensión de los textos [Online]. Available: https://hulat.inf.uc3m.es/noticia/PlataformaEASIERunaayudaenlacomprensi%C3%B3ndelostextos

[5] Centro Español de Accesibilidad Cognitiva. (2025). Aplicación para adaptar textos a lectura fácil [Online]. Available: https://www.ceacog.es/que-hacemos/aplicacion-para-adaptar-textos-a-lectura-facil/

[6] M. Cañete *et al.*, "Spanish pre-trained BERT model and evaluation data," *arXiv preprint*, arXiv:1907.11692, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[7] Y. Liu *et al.*, "RoBERTa: A robustly optimized BERT pretraining approach," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Florence, Italy, 2019, pp. 1–6. [Online]. Available: https://aclanthology.org/N19-1423/

[8] R. Maldonado. (2025, Apr. 15). Ventajas y desventajas de Python [Online]. Available: https://keepcoding.io/blog/ventajas-y-desventajas-de-python/ Hugging Face

[9] spaCy. (n.d.). Doc [Online]. Available: https://spacy.io/api/doc spacy.io

[10] spaCy. (n.d.). Spanish models. [Online]. Available: https://spacy.io/models/es

[11] spaCy. (n.d.). Matcher [Online]. Available: https://spacy.io/api/matcher Kaggle

[12] Hugging Face. (n.d.). Transformers [Online]. Available: https://huggingface.co/docs/transformers/index

[13] A. Gutiérrez Fandiño *et al.*, "MarIA: Spanish language models," *Procesamiento del Lenguaje Natural*, no. 68, 2022. [Online]. Available: https://doi.org/10.26342/2022-68-3

[14] A. Gonzalez-Agirre *et al.*, "Salamandra technical report," *arXiv preprint*, arXiv:2502.08489, 2025. [Online]. Available: https://arxiv.org/abs/2502.08489

[15] IBM. (n.d.). *¿Qué es PyTorch?* [Online]. Available: https://www.ibm.com/es-es/topics/pytorch

[16] Python Software Foundation. (n.d.). *configparser — Configuración de archivos INI*. [Online]. Available: https://docs.python.org/es/3.13/library/configparser.html

[17] Arsys. (n.d.). *¿Qué es Visual Studio Code y cuáles son sus ventajas?* [Online]. Available: https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas

[18] J. Alphonso. (n.d.). Filling in <masked> words with RoBERTa [Online]. Available: https://www.kaggle.com/code/juliusalphonso/filling-in-masked-words-with-roberta

[19] Barcelona Supercomputing Center. (n.d.). ALIA Kit: Inicio [Online]. Available: https://langtech-bsc.gitbook.io/alia-kit

[20] Adjetivos.org. (n.d.). *Adjetivos sustantivados*. [Online]. Available: https://adjetivos.org/adjetivos-sustantivados/

[21] Lifeder. (n.d.). *Sustantivación de adjetivos: qué es, ejemplos y oraciones*. [Online]. Available: https://www.lifeder.com/sustantivacion-de-adjetivos/

[22] Lenguaje.com. (n.d.). *Sintagma nominal: ejercicios resueltos*. [Online]. Available: https://lenguaje.com/sintagma-nominal-ejercicios/

[23] P. Gómez Muñoz, "La sustantivación en español: Perspectiva gramatical y discursiva," *Revista Signos. Estudios de Lingüística*, vol. 39, no. 61, pp. 49–69, 2006. [Online]. Available: https://revistasignos.cl/index.php/signos/article/view/224/198

7 Anexos

Anexo 1: Estructura modular del repositorio

Se ha seguido una organización modular y clara, con el objetivo de facilitar tanto el desarrollo progresivo como el mantenimiento y reutilización del código. A continuación, se muestra la organización que siguen los archivos dentro del proyecto.

Figura 2: Situación de la estructura del repositorio a lo largo de las ejecuciones

Las imágenes de la Figura 2 representan el estado de la estructura del programa previo a su ejecución, tras la detección y tras la adaptación, respectivamente.

A continuación, se describe la función de los directorios y archivos más relevantes:

Archivos de configuración y documentación

README.md: archivo de documentación principal. Contiene una descripción general del proyecto, instrucciones de uso y ejemplos de ejecución. Sirve como guía rápida para cualquier persona que consulte el repositorio por primera vez.

config.ini: archivo de configuración centralizado. Contiene parámetros clave que el sistema necesita para funcionar correctamente, como el modelo de spaCy a utilizar o la ruta a los ficheros de entrada/salida. Facilita la adaptación del código sin necesidad de modificar directamente los módulos.

.gitignore: indica a Git qué archivos o carpetas deben ser excluidos del control de versiones (por ejemplo, el entorno virtual o ficheros temporales).

requirements.txt: lista de dependencias necesarias para ejecutar el proyecto. Permite instalar fácilmente todos los paquetes necesarios mediante un simple comando (pip install -r requirements.txt).

Carpeta .venv/

Esta carpeta contiene el entorno virtual de Python, que aísla las dependencias del proyecto del sistema global. Así se asegura la compatibilidad y portabilidad del entorno de ejecución. Esta carpeta no se sube al repositorio gracias al *.gitignore*, ya que puede regenerarse en cualquier máquina.

Carpeta data/

Esta carpeta almacena los ficheros de entrada y salida del sistema. Es esencial para la ejecución y validación del proyecto. Contiene:

textos.csv: archivo base con más de 300 frases de prueba. Incluye ejemplos con y sin adjetivos sustantivados. Ha sido el corpus principal utilizado durante el desarrollo, pruebas y validación del sistema.

adaptar_mask.csv: archivo generado tras la detección de adjetivos sustantivados marcados con <mask>. Contiene frases con esas marcas insertadas, y sirve como entrada para el módulo de adaptación automática.

mask_adaptadas.csv: resultado de aplicar la adaptación a las frases con <mask>. Incluye la frase original y dos versiones adaptadas.

adaptar_lo.csv y *lo_adaptadas.csv*: ficheros equivalentes a los anteriores, pero específicamente para los casos en los que el adjetivo sustantivado está precedido por "lo", que requieren un tratamiento específico.

Carpeta nlp/

Este directorio contiene el núcleo funcional del sistema: los módulos de procesamiento lingüístico. Cada archivo cumple una función específica:

config.py: carga y gestiona los parámetros definidos en config.ini. Se importa en el resto de los módulos para acceder a configuraciones globales, como el modelo de spaCy o listas de palabras relevantes.

detection.py: módulo encargado de la detección de adjetivos sustantivados. Define el pipeline personalizado de spaCy y contiene los patrones lingüísticos utilizados para identificar construcciones sustantivadas.

adaptation.py: módulo que implementa la adaptación de las frases marcadas con <mask> o "lo". Propone versiones más explícitas utilizando tanto reglas lingüísticas como modelos de lenguaje.

pipeline_base.py: clase base compartida por los distintos componentes del pipeline. Sirve como esqueleto común y puede facilitar extensibilidad o reutilización del código.

utils.py: funciones auxiliares generales, como la escritura en CSV o el preprocesamiento de textos. Centraliza operaciones comunes que no pertenecen a un módulo específico.

Carpeta scripts/

Contiene scripts de ejecución que actúan como puntos de entrada al sistema. Están diseñados para facilitar el uso del sistema por parte del desarrollador o de usuarios no técnicos:

run_detection.py: ejecuta el pipeline de detección sobre el archivo textos.csv y genera como resultado *adaptar_mask.csv* y/o *adaptar_lo.csv*.

run_adaptation.py: toma los archivos de detección y produce las frases adaptadas (mask_adaptadas.csv y lo_adaptadas.csv), aplicando las estrategias descritas en los módulos.

Esta organización modular y jerárquica ha permitido separar claramente las responsabilidades de cada parte del sistema (detección, adaptación, configuración, ejecución), favoreciendo el desarrollo incremental y facilitando tanto las pruebas como la documentación del proyecto.

Las partes del sistema más robustas y con mayor relevancia son las de descripción y adaptación, por lo que serán descritas en profundidad en los próximos apartados.

Anexo 2: Resultados de Adaptación 'lo' + adjetivo

Frase original	**Frase adaptada Salamandra**
Lo más bonito de Barcelona es la Sagrada Familia.	Las mejores cosas que ver en Barcelona son la Sagrada Familia y el Parc Güell.
Lo bueno es que aún tienes otra oportunidad.	Es bueno que todavía tengas una oportunidad más.
Lo mejor aún está por venir.	La mejor parte todavía no ha llegado.
Lo bueno de la vida	La buena parte en la existencia
Lo malo de la vida	La mala parte de vivir es…
Los romanos respetaban lo antiguo	Los romanos respetaban lo viejo
Lo dulce de su mirada.	La dulzura en su mirada.
Lo nuevo no supera lo antiguo	La nueva no es superior al viejo.
A Verónica le gusta lo raro, lo excepcional.	A Verónica le gustan los raros y excepcionales.
En su obra lo español está siempre presente.	Su obra tiene una presencia constante de lo español".
Lo más bonito de esta ciudad es su arquitectura	La mejor parte de la ciudad es su arquitectura.
Gracias por venir, y disculpen lo malo	Gracias por venir; y disculpe las molestias
Verdaderamente, lo raro es verte por aquí.	Verdaderamente, extraño verte por aquí.
Lo pequeño es hermoso, siempre y cuando sea caro.	La pequeñez es bella, siempre que cueste un dineral.
Según Rosario, ahora es que viene lo bueno.	Según Rosario, ahora es cuando viene la parte interesante.
Lo inesperado surge precisamente cuando nadie lo espera.	Inesperadas son aquellas cosas que suceden en un momento imprevisto.
Lo bueno siempre se hace esperar.	La buena comida siempre es esperada con ansias.
Lo ideal es no esperar nada.	Es mejor no tener expectativas.
Lo intrigante de tu mirada es motivo de alabanza.	Tu mirada intrigante es motivo de alabanza.
Lo horroroso de mi infancia	La horribleidad de mi niñez
Lo cotilla que es tu madre	Tu madre es una chismosa
Lo útil nunca sera desperdiciado	Nunca será desperdiciada la utilidad.

Anexo 3: Recibo del Informe de Originalidad 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:	PAULA CASTILLO GARCIA
Título del ejercicio:	Turnitin Memoria Final
Título de la entrega:	Memoria_Final_TFG_Paula_Castillo.pdf
Nombre del archivo:	13811_PAULA_CASTILLO_GARCIA_Memoria_Final_TFG_Paula_C...
Tamaño del archivo:	1.44M
Total páginas:	55
Total de palabras:	14,345
Total de caracteres:	83,646
Fecha de entrega:	04-jun.-2025 10:25a. m. (UTC+0200)
Identificador de la entrega:	2691902410

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

LECTURA FÁCIL: FRASES NOMINALES y USOS NOMINALES de ADJETIVOS

Autor: Paula Castillo García
Tutor: María del Carmen Suárez de Figueroa Baonza
Cotutor: Isam Diab Luzano

Madrid, junio de 2025

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jun 04 15:59:02 CEST 2025
	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)