Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Lectura Fácil: Relación entre Ideas

Autor: Sergio Cabello Moralo

Tutor(a): Mari Carmen Suárez de Figueroa Baonza

Madrid, enero de 2023

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 Ingeniería Informática
Título: Lectura Fácil: Relación entre Ideas
Enero 2023

Autor: Sergio Cabello Moralo

Tutor:
Mari Carmen Suárez de Figueroa Baonza
Departamento de Inteligencia Artificial (DIA)
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Uno de los principales impedimentos para acceder al conocimiento o la cultura para algunas personas es la complejidad de los textos en cuestión, donde muchas veces una estructura muy compleja o la utilización de cultismos y modismos dificulta en gran medida su comprensión.

La metodología de *Lectura Fácil* proporciona una serie de pautas y recomendaciones al respecto sobre cómo debería ser un texto totalmente accesible que pueda ser comprendido por cualquier persona. Estas pautas están orientadas a la simplificación de textos para hacerlos más accesibles para personas más vulnerables en este sentido: aquellas que tengan alguna discapacidad cognitiva, que sufran algún trastorno del lenguaje como la afasia o que estén aprendiendo el idioma.

Este trabajo se centra en el desarrollo de un software que, en primer lugar identifique un tipo muy concreto de oraciones sintácticamente complejas (subordinadas adverbiales causales externas) y después transforme dichas estructuras a otras que sean más sencillas sintácticamente pero que mantengan el significado de la oración original.

Este trabajo se integrará en un proyecto común al que han aportado otros muchos alumnos en sus correspondientes TFG, TFM o tesis con el fin de crear una herramienta lo más completa posible que sea capaz de identificar y transformar muchos patrones complejos en textos reales y sustituirlos por otros que sigan las recomendaciones que marca la metodología de Lectura Fácil.

Abstract

One of the main impediments to accessing knowledge or culture for some people is the complexity of that texts, where often a very complex structure or the use of literary words and idioms greatly hinders their understanding.

The Easy2Read methodology provides a series of guidelines and recommendations in this regard on how a fully accessible text that can be understood by anyone should be. These guidelines are aimed at simplifying texts to make them more accessible for vulnerable people in this sense: those who have a cognitive disability, who suffer from a language disorder such as aphasia or who are learning the language.

This project focuses on the development of software that firstly identifies a very specific type of syntactically complex sentences (external causal adverbial subordinates) and then transforms these structures into others that are syntactically simpler but maintain the meaning of the original sentence.

This project will be integrated into a common project to which many other students have contributed in their corresponding TFG, TFM or thesis in order to create a complete tool that is capable of identifying and transforming many complex patterns in real texts and replacing them with others that follow the recommendations established by the Easy2Read methodology.

Tabla de contenidos

1 Introducción

Uno de los canales de comunicación más frecuentes e importantes es la escritura, puesto que permite la transmisión de información de manera literal y sin los problemas de deformación del mensaje original que puede tener la comunicación oral. Sin embargo, hay personas para las que, dada la complejidad de ciertos textos, esto se puede complicar bastante, ya sea porque tienen alguna condición particular que les dificulta la lectura (discapacidades cognitivas, trastornos del lenguaje, etc.) o porque están aprendiendo el idioma entre otras.

Esto supone una barrera para un gran número de personas a la hora de acceder a todo tipo de conocimiento, desde textos técnicos o académicos hasta textos literarios, noticias o documentación. Por tanto en caso de tener alguna condición que dificulte la comprensión lectora, una persona puede verse lastrada para acceder a una formación académica, a diversas formas de entretenimiento y en general, a cualquier tipo de información que se transmita por escrito.

Para afrontar estas cuestiones, la metodología de *Lectura Fácil* [1] propone una serie de recomendaciones y pautas que debe seguir un texto para ser tan accesible como sea posible.

El problema reside en que la transformación de escritos en base a estas recomendaciones puede ser complicada si se hace manualmente dado el gran trabajo que conlleva y la ambigüedad que puede surgir debido a diferencias en el criterio para aplicar transformaciones sobre un mismo texto por distintas personas.

Esto crea la necesidad de tener alguna herramienta que permita, en primer lugar poder transformar textos en base a las pautas que proporciona esta metodología, además de tener un criterio unificado para aplicar estas sustituciones.

El presente TFG trabaja en esta línea al igual que el resto de trabajos y tesis que se han desarrollado en la línea de investigación a la que pertenece. En general se trata de identificar algún tipo de incidencia en un texto que no siga las pautas que marca Lectura Fácil y la sustitución de dichos fragmentos de texto por otros que sean semánticamente similares pero con una estructura más simple que siga las recomendaciones de esta metodología.

En concreto en este trabajo se pretende identificar un tipo de oraciones subordinadas (que de por si son estructuras sintácticas complejas), las adverbiales causales externas y sustituirlas por oraciones simples que mantengan el significado original.

En el Capítulo 2 se ahonda en algunos detalles sobre los trabajos previos de esta línea de investigación, sobre la metodología de Lectura Fácil y sobre las tecnologías utilizadas para el desarrollo del TFG, incidiendo principalmente en la librería spaCy [2]. También, en el Capítulo 3 se explica en detalle la implementación de las pautas de identificación y adaptación así como las pruebas realizadas para comprobar la efectividad y fiabilidad de los servicios.

2 Estado del Arte

En este capítulo se describe brevemente la línea de investigación a la que pertenece este TFG, dando una idea de los aspectos que se han tratado en trabajos anteriores y proporcionando algunos ejemplos y también se ahonda en algunos conceptos importantes en el desarrollo del trabajo, siendo estos la metodología de Lectura Fácil, el procesamiento del lenguaje natural como uno de los campos de investigación más punteros en el ámbito de la Inteligencia Artificial y haciendo hincapié en el la librería de procesamiento de lenguaje natural spaCy, que ha sido la principal herramienta utilizada en este trabajo.

2.1 Trabajos previos

Este trabajo al igual que sus predecesores forma parte de una línea de investigación que trata de crear un proyecto formado por pautas independientes, cada una de las cuales aplica de forma automática alguna de las recomendaciones que especifica la metodología de Lectura Fácil. En la mayoría de los casos detectando incidencias y sustituyéndolas por fragmentos de texto que sigan dichas pautas. El objetivo a futuro es tener una aplicación completa que agrupe todas estas pautas y permita transformar textos automáticamente para que cumplan con lo que marca la metodología.

Para la elaboración de esta memoria se ha tenido en cuenta la estructura de otros trabajos pertenecientes a la misma línea de investigación con el fin de incluir la información más relevante y mantener una coherencia entre los distintos trabajos relacionados.

Los trabajos que se han tenido en cuenta son los siguientes:

- Gil de Eusebio, Guillermo – Lectura Fácil: Adaptador de Texto [3]
- Ming Díaz, Alejandro – Lectura Fácil: Perífrasis Verbales [4]
- Antona Palacios, Julia – Lectura Fácil: Perífrasis Verbales 2.0 [5]

2.2 Metodología Lectura Fácil

Tal y como se describe en el documento descriptivo de la metodología, “la lectura fácil surge como una herramienta de comprensión lectora y de fomento de la lectura para atraer a personas que no tienen hábito de leer o que se han visto privadas de él”.

Es decir que la idea que persigue Lectura Fácil es la de democratizar el acceso a todo tipo de información, cultura o entretenimiento en soporte escrito, dado que es un derecho fundamental de las personas y como tal debe garantizarse para cualquier persona, independientemente de sus capacidades.

Esto es importante ya que la incapacidad que pueda tener una persona para comprender un texto le inhabilita no solo para acceder a textos literarios o académicos, sino también para acceder a textos de la vida cotidiana tales como documentos administrativos, informes médicos, notificaciones bancarias, legislación, etc. Lo cual puede crear unas enormes diferencias sociales a la larga que derive en una disgregación de las sociedades, donde existan ciudadanos “de primera” y “de segunda” en función de sus capacidades. Esta problemática es la que se pretende atajar con la metodología de Lectura Fácil. En el documento descriptivo de lectura fácil referenciado en el capítulo 7 [1] se pueden encontrar referencias legislativas relacionadas con Lectura Fácil y un capítulo dedicado a describir y enumerar los tipos de usuarios hacia los que está dirigida la metodología.

En cuanto a la aplicación o las pautas que propone Lectura Fácil las divide en áreas de aplicación, las cuales son: Redacción, Diseño y Maquetación, Producción, Otros y Anotaciones para Relatos.

En el área de Diseño y Maquetación se incluyen recomendaciones acerca de la tipografía utilizada, las ilustraciones e imágenes que aparecen en el texto y demás cosas relacionadas con el aspecto y la parte visual del texto. En Producción se habla de aspectos relacionados con el soporte, tales como las características del papel a utilizar, la recomendación de encuadernar las obras o el código de colores utilizado. En Otros aparecen recomendaciones que no se pueden incluir en las anteriores categorías por ser algo más específicos y relacionados con la ayuda interpersonal para la comprensión de textos de Lectura Fácil y por último en el área de Anotaciones para Relatos se habla de aspectos específicos para los textos literarios, con recomendaciones acerca de cómo deben identificarse los diálogos por escrito, la cantidad y complejidad de los personajes, el estilo narrativo, etc.

No se ha mencionado el área de redacción anteriormente porque se va a explicar con algo más de detalle a continuación, dado que es el área de recomendaciones en la que se mueve este TFG.

En el área de la redacción se han establecido las siguientes categorías:

- ORTOGRAFÍA (Ej. Uso de mayúsculas, signos de puntuación, fechas, etc.)
- GRAMÁTICA (Ej. Uso de ciertos tiempos y modos verbales, oraciones complejas o figuras que puedan crear confusión como aposiciones o la elisión del sujeto)
- LÉXICO (Ej. Complejidad del vocabulario, uso de palabras polisémicas, acrónimos, abreviaturas, etc.)
- ESTILO (Ej. Intentar ser minimalista a la hora de expresar las ideas, evitando contenido redundante o poco importante)

El desarrollo de este trabajo se centra en implementar una pauta referente a la categoría de GRAMÁTICA. Las oraciones subordinadas, tanto las causales en este caso como el resto, son estructuras sintácticas complejas que en ocasiones dificultan la comprensión de los textos. Es por eso que el propósito de este TFG es identificar dichas estructuras (subordinadas causales) y transformarlas a oraciones simples.

2.3 Procesamiento del Lenguaje Natural: spaCy

En primer lugar hay que definir brevemente el concepto de Procesamiento del Lenguaje Natural o NLP por sus siglas en inglés.

Se trata de una de las ramas de conocimiento pertenecientes a la inteligencia artificial. Su propósito es estudiar las estructuras que conforman el lenguaje natural, las palabras que forman un idioma y la forma en que se conectan y relacionan entre ellas con el fin de reproducir o reconocer de forma automática, mediante software, uno o varios idiomas como lo haría un humano. Es una línea de investigación muy abierta en el campo de la inteligencia artificial pero que avanza rápido, consiguiendo resultados impensables hace unos años.

La forma en la que estas inteligencias artificiales procesan texto tiene principalmente dos vertientes: a través de mecanismos basados en reglas o patrones que aplican lógica clásica de programación, o mediante redes neuronales, es decir, modelos basados en el Aprendizaje Automático o Machine Learning [6] (ML por sus siglas en inglés) que analizan las relaciones estadísticas

que existen en un sistema, en este caso un lenguaje humano, y aprenden de ellas para poder realizar diversas tareas.

En el capítulo 7 se incluyen un par de referencias sobre los conceptos de red neuronal y procesamiento del lenguaje natural [7][8].

El procesador de lenguaje natural utilizado en el desarrollo de este trabajo, spaCy, es un caso de esto último, basado en ML. La razón de escoger spaCy sobre otras librerías de procesamiento de lenguaje se debe al buen funcionamiento que tiene para el español y a la velocidad de procesamiento.

Finalmente vamos a explicar un concepto importante para entender el trabajo que realizamos con spaCy, este concepto es el PoS tagging (Part of Speech tagging).

El PoS tagging [9][10] es un proceso en el que se etiqueta cada palabra presente en un texto con diferentes atributos referentes a varios aspectos de esa palabra en relación con la totalidad del texto, algunas de estas son el tipo de palabra, las dependencias sintácticas, información morfológica, entre otras.

Etiquetas PoS existentes en spaCy:

POS	DESCRIPTION	EXAMPLES
ADJ	adjective	big, old, green, incomprehensible, first
ADP	adposition	in, to, during
ADV	adverb	very, tomorrow, down, where, there
AUX	auxiliary	is, has (done), will (do), should (do)
CONJ	conjunction	and, or, but
CCONJ	coordinating conjunction	and, or, but
DET	determiner	a, an, the
INTJ	interjection	psst, ouch, bravo, hello
NOUN	noun	girl, cat, tree, air, beauty
NUM	numeral	1, 2017, one, seventy-seven, IV, MMXIV
PART	particle	’s, not,
PRON	pronoun	I, you, he, she, myself, themselves, somebody
PROPN	proper noun	Mary, John, London, NATO, HBO
PUNCT	punctuation	., (,), ?
SCONJ	subordinating conjunction	if, while, that
SYM	symbol	$, %, §, ©, +, −, ×, ÷, =, :), 😡
VERB	verb	run, runs, running, eat, ate, eating
X	other	sfpksdpsxmsa
SPACE	space	

1 Etiquetas POS de spaCy [11]

Por último, debe entenderse correctamente la estructura de objetos que conforman la librería spaCy:

2 Arquitectura librería spaCy [12]

El objeto más importante de la arquitectura y el que recibirán las funciones implementadas en el desarrollo del trabajo es el objeto Doc, este objeto contiene una lista de tokens que son el resultado del proceso del Part-of-Speech Tagging que realiza la librería, este objeto tiene una serie de atributos a los que podemos acceder y modificar a nuestro antojo (doc.pos_, doc.dep_, doc.lemma, doc.text_, doc.morph_, entre otros). Detalles sobre estos atributos de desgranan en el capítulo 3.

2.4 Subordinación causal externa

En primer lugar es necesario entender lo que es una oración subordinada, explicado de forma sencilla diríamos que es una oración compuesta, en la que existe una estructura que funciona como oración auxiliar y que depende de una oración principal. Es decir, son oraciones en las que existe una estructura dependiente de la oración principal cuya función es complementar el significado de la misma (sin la oración principal no tienen sentido).

Por tanto, una oración subordinada puede expresar lo mismo que una oración simple pero es sintácticamente más compleja.

Existen varios tipos de oraciones subordinadas dependiendo de su función sintáctica pero en este caso las que nos interesan para entender el desarrollo de este trabajo son las subordinadas adverbiales que funcionan como un adverbio o grupo adverbial. Dentro de este tipo de subordinadas también podemos distinguir entre varios subtipos, uno por cada función que pueda cumplir un adverbio en una oración. Este TFG se centra en las subordinadas adverbiales causales, es decir, las que implican una relación causa-efecto.

Este tipo de estructuras tienen una estructura bastante fija que se compone de la conjunción “como” situada al inicio de una oración con un verbo en modo

indicativo seguido de una oración simple que es la principal y suele actuar como efecto en la relación causa-efecto mencionada. [13][14]

Por tanto en la fase de detección se deben tener en cuenta las siguientes características: una oración que comience por la palabra “como” actuando como nexo, un verbo en modo indicativo que forme parte de una cláusula adverbial, y por último un verbo que actúe como raíz de la oración y aparezca después de la cláusula adverbial (para confirmar que estamos en una subordinación y que es causal).

3 Desarrollo

Es recomendable comentar brevemente la estructura del proyecto para evitar confusiones en los apartados posteriores.

El proyecto, que es común para todos los trabajos desarrollados por los distintos alumnos que han participado en proyectos de esta línea de investigación, se compone de una estructura de paquetes, uno por cada servicio ofrecido, siendo estos *contentExtractionService*, *identificationService*, *adaptationService*, *suggestionService* y *statusService*. Estos servicios realizan funciones auxiliares a lo referente a este TFG por lo que no entraremos en profundidad a explicarlos, simplemente es conveniente saber que son servicios que facilitan la integración de las pautas implementadas.

La pauta desarrollada para este TFG así como las demás se ubican dentro del proyecto *commos*, del cual cuelgan otros dos directorios referentes a diferentes tipos de pautas, estos son *designGuidelines* y *writingGuidelines*. La pauta que nos concierne forma parte del proyecto *writingGuidelines*, del cual a su vez cuelgan tres directorios en función de la característica que intenten detectar o sustituir, estos son *orthographyGuidelines*, *sentenceGuidelines*, *styleGuidelines* y *vocabularyGuidelines*.

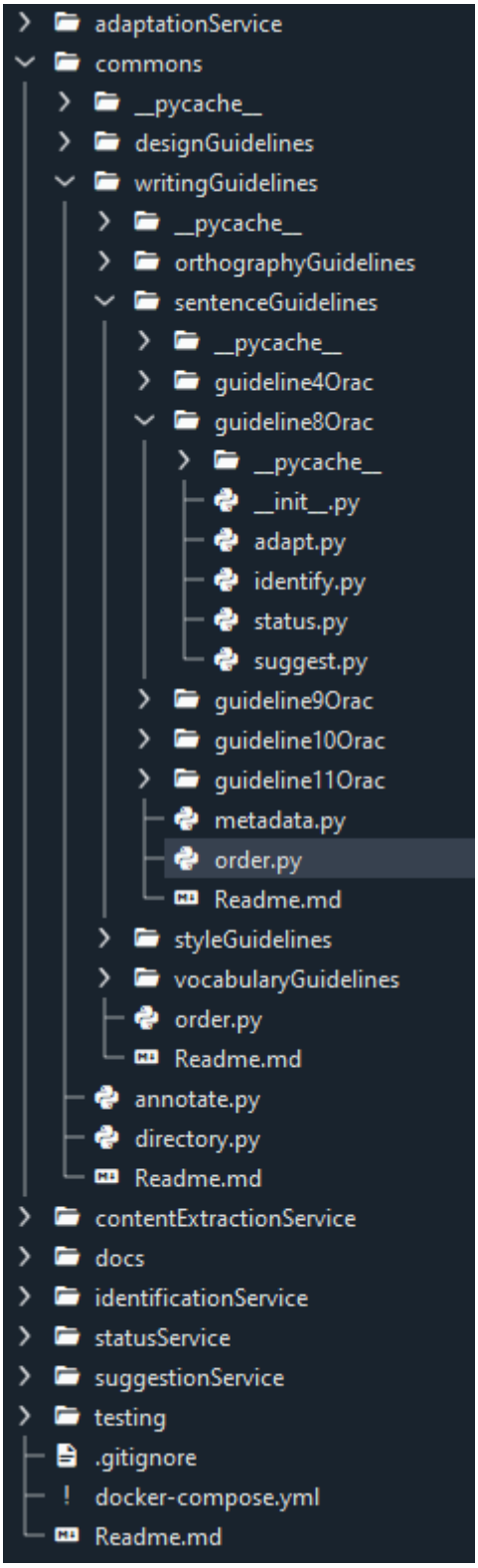
En nuestro caso es de tipo *sentenceGuidelines*. En este directorio se encuentran todas las pautas de este tipo que ya han sido implementadas. La que concierne a este trabajo es *guideline8Orac*.

Cada pauta de este tipo se compone de una serie de ficheros que implementan las distintas funciones de dicha pauta, estas son identify.py, adapt.py, suggest.py y status.py. Se comentan más adelante.

Por último existe otro paquete llamado *testing*, que cuelga del directorio raíz del proyecto en el que se implementan los testers de las pautas implementadas.

En cuanto al problema a resolver tenemos un tipo de oraciones sintácticamente complejas como son las subordinadas causales, que expresan una relación de causalidad entre la oración auxiliar y la principal que la componen. Estas estructuras son unas de las identificadas como problemáticas por Lectura Fácil, que propone como alternativa la conexión de las dos ideas que forman parte de esta subordinación (causa y efecto) mediante dos oraciones simples separadas por un punto y un nexo, consiguiendo dos estructuras simples en lugar de una compleja que expresen la misma relación de causalidad y mantengan el sentido de la frase original.

En este trabajo se implementa la Pauta 4 de la sección 6.1 de la norma UNE: “Las ideas enlazadas se deberían separar mediante un punto en lugar de una coma”. [15][16]



3 Estructura de ficheros del proyecto

3.1 Fase de identificación

La primera fase es la de identificación (implementada en identify.py) cuyo propósito es detectar si en un texto existe una oración subordinada causal en base al etiquetado PoS realizado por spaCy. Esto se consigue identificando las características definitorias de este tipo de oraciones a través de dicho etiquetado. ¿Pero qué se busca exactamente?

En primer lugar este tipo de oraciones siempre van a venir precedidas de un adverbio ya que es un caso de subordinación adverbial (de causa). En nuestro caso siempre será el mismo: como.

Por tanto el primer patrón a identificar es la aparición de la palabra **como** al inicio de una oración, además tras el etiquetado de los tokens que hace spaCy, esta debe ir acompañada de la etiqueta "SCONJ" (sintagma conjuntivo) y "mark" (marker o nexo). Si detectamos esto podemos saber que estamos en una oración subordinada pero debemos comprobar de qué tipo.

Buscamos subordinadas adverbiales causales por lo que el siguiente paso será la identificación de un verbo (etiqueta "VERB") que sea una cláusula adverbial (etiqueta "advcl") y que esté en modo indicativo (etiqueta "Mood=Ind"). Se pueden encontrar casos en los que estas etiquetas estén repartidas entre dos o más palabras que formen una perífrasis verbal o que todas estén relacionadas con una sola palabra, un verbo. Ambos casos son factibles. Si se consigue detectar este segundo patrón todo indica que estamos ante una subordinada adverbial causal pero debemos verificarlo.

Para comprobar que los indicios anteriores están marcando la existencia de una subordinada adverbial causal se debe verificar la existencia de la otra parte de una subordinada, la oración principal. Para esto debemos buscar un verbo que esté marcado con las etiquetas "VERB" y "ROOT" (que sea un verbo y que además sea raíz o verbo principal).

Si se han detectado estos tres patrones todo indica que estamos ante una subordinada adverbial causal, sin embargo existen ejemplos de oraciones que no son subordinadas causales que siguen la misma estructura sintáctica que estas y por tanto, spaCy identificará en ella los mismos patrones, lo que nos puede llevar fácilmente a la detección de falsos positivos. Para solucionar este problema se planteó el uso de otro procesador de lenguaje natural que analiza la distancia semántica entre dos oraciones para comparar conjuntos de oraciones subordinadas causales y sus transformaciones y por otro lado hacer lo mismo con oraciones que no eran subordinadas causales. Las conclusiones y los resultados acerca de esto se explican en el capítulo de Pruebas (3.3)

¿Cómo se implementa esta funcionalidad? Aquí van unos detalles al respecto:

El primer paso sería instalar la librería spaCy y el pipeline entrenado para trabajar con textos en español [17].

Una vez instalada ya podemos empezar a trabajar con spaCy y existen varias opciones posibles para detectar la estructura que buscamos. En un primer momento se valoró utilizar el objeto Matcher [18] proporcionado por la arquitectura de spaCy, que permite buscar patrones en un texto en base a estructuras similares a un diccionario o un objeto JSON, en el cual se establecen pares clave-valor en los que se pueden especificar características que se desean encontrar en una cadena de tokens. Estas características pueden referirse al tipo de palabra (atributo POS), dependencias (atributo DEP), etc. El problema que surgía al utilizar esta estrategia es que el uso de patrones con un

objeto Matcher requiere de un conocimiento muy preciso sobre las estructuras que se quieren parsear en el texto, ya que se debe especificar qué tokens y en qué orden se deben buscar. Esto era un problema insalvable para la detección de subordinadas causales externas puesto que, aunque su estructura es bastante fija, no podemos saber el número de tokens que la compondrán ni las estructuras que se pueden intercalar con ellas. Solamente sabemos que debemos detectar las características descritas más arriba y en el orden especificado, pero entre medias podría aparecer cualquier cosa y la estructura que buscamos seguiría siendo correcta. Como no hay manera de solventar esto con patrones (ni siquiera con el elemento comodín que se puede introducir en ellos) se descartó esta solución y se optó por la implementación de un método declarativo basado en reglas.

En cuanto al código de la función de identificación, esta recibe tres parámetros de entrada:

- text -> La cadena de texto a analizar
- format_information -> Un parámetro opcional que da información adicional sobre el texto
- document -> El objeto que contiene los tokens etiquetados por spaCy

```
def identify(text, format_information, document):
```

4 Firma o signature de la función identify

En el inicio de la función se instancia una lista vacía que posteriormente se rellenará con información sobre las incidencias, es decir, subordinadas causales que se hayan encontrado en el texto si es que se detecta alguna y se retornará como parámetro de salida.

```
issues = []
```

5 Parámetro de salida de la función

La función se estructura entorno a un bucle que se repite para cada sentencia u oración presente en el texto, esto se hace a través del iterable que devuelve la propiedad *sents* [19] del objeto *document*.

```
for sentence in document.sents:
```

6 Iterable recorrido por la función identify

Tras una comprobación de que la sentencia a analizar no sea una cadena vacía, se inicializan las estructuras de datos correspondientes para realizar la lógica de la función. Las cuatro primeras (*verb_detected*, *advcl_detected*, *ind_detected* y *root_detected*) son listas vacías al inicio en las que se guardarán los índices de los tokens en los que se detecte el etiquetado sintáctico correspondiente a un verbo (“VERB”), un verbo cuyas dependencias indiquen que forma parte de una cláusula adverbial (“advcl”), un verbo cuyas dependencias indiquen que es el verbo raíz o principal de una oración (“ROOT”) o un verbo en modo indicativo (“Mood=Ind”), respectivamente. En caso de no detectarse ningún caso de alguno o de todos los mencionados en el texto a analizar, las listas permanecerían vacías. Y al ser necesaria la aparición de al menos una ocurrencia de cada uno de ellos para que exista la subordinación causal, esto haría que para esa sentencia no se reflejase ninguna incidencia en el parámetro de salida. Cabe destacar que estas etiquetas que se buscan en el texto pueden aparecer varias de ellas a la vez en un mismo token, lo importante es el orden de ocurrencia a

la hora de saber si es un caso de subordinación causal, lo cual se detalla más adelante.

```
verb_detected = []
advcl_detected = []
ind_detected = []
root_detected = []
```

7 Listas auxiliares

Después se inicializa un flag que utilizaremos en el final de la función para decidir si añadir una entrada al parámetro de salida en función de si se ha detectado subordinación causal para la sentencia que se está analizando en esa iteración. Y por último cinco listas que se inicializan con parámetros provenientes del objeto document de spaCy, que se recibe como parámetro de entrada. Estas listas son: una lista con todos los tokens de la sentencia (*tokens*); una lista con los valores de la variable lemma de cada token (*lemmas*), esta variable contiene la "palabra raíz" de la que aparece en el texto del token, en el caso de "niña" o "niños", sería "niño"; una lista con la categoría gramátical de cada palabra (*t_pos*); las dependencias sintácticas de cada token (*deps*); y la información morfológica adicional de cada token (*morph*).

```
advcl_before_root = False
tokens = [t for t in sentence]
lemmas = [t.lemma_ for t in sentence]
t_pos = [t.pos_ for t in sentence]
deps = [t.dep_ for t in sentence]
morph = [t.morph for t in sentence]
```

8 Estructuras de datos del programa

Tras la inicialización de las variables de control se comienza a analizar la sentencia. Se hace una distinción entre las oraciones que empiezan con un signo de puntuación como las preguntas o las exclamaciones y las que no, en caso que de empiece por un signo de puntuación se busca la palabra "como" en el segundo token y en caso contrario en el primero. Por lo demás la lógica en ambos casos es idéntica, a parte de esta pequeña distinción en la primera condición también se comprueba que la categoría gramatical del primero o segundo token (en función de la distinción del inicio) sea sintagma conjuntivo ("SCONJ") y que en sus dependencias aparezca la etiqueta de nexo ("mark").

Si se cumplen las respectivas condiciones se itera sobre la lista de tokens para buscar ocurrencias del resto de características propias de la subordinación causal.

```
if tokens[0].is_punct:
    if lemmas[1] == "como" and t_pos[1] == "SCONJ" and deps[1] == "mark":

        for i, item in enumerate(tokens):

            x = re.search("VERB", str(t_pos[i]))
            if x is not None:
                verb_detected.append(i)

            x = re.search("advcl", str(deps[i]))
            if x is not None:
                advcl_detected.append(i)

            x = re.search("ROOT", str(deps[i]))
            if x is not None:
                root_detected.append(i)

            x = re.search("Mood=Ind", str(morph[i]))
            if x is not None:
                ind_detected.append(i)

        if len(verb_detected) > 0 and len(advcl_detected) > 0 and len(ind_detected) > 0 and len(root_detected) > 0:
            for tupla in zip(advcl_detected, root_detected):
                if tupla[0] < tupla[1]:
                    advcl_before_root = True
                    break
else:
    if lemmas[0] == "como" and t_pos[0] == "SCONJ" and deps[0] == "mark":

        for i, item in enumerate(tokens):

            x = re.search("VERB", str(t_pos[i]))
            if x is not None:
                verb_detected.append(i)

            x = re.search("advcl", str(deps[i]))
            if x is not None:
                advcl_detected.append(i)

            x = re.search("ROOT", str(deps[i]))
            if x is not None:
                root_detected.append(i)

            x = re.search("Mood=Ind", str(morph[i]))
            if x is not None:
                ind_detected.append(i)

        if len(verb_detected) > 0 and len(advcl_detected) > 0 and len(ind_detected) > 0 and len(root_detected) > 0:
            for tupla in zip(advcl_detected, root_detected):
                if tupla[0] < tupla[1]:
                    advcl_before_root = True
                    break
```

9 Flujo principal de ejecución

Si se cumplían las condiciones iniciales se itera sobre la lista de tokens y se buscan las siguientes coincidencias mediante expresiones regulares:

- La etiqueta “VERB” en la lista de categorías gramaticales de los tokens (*t_pos*). En caso de haber coincidencia se guarda el índice del token correspondiente en la lista *verb_detected*.
- La etiqueta “advcl” en la lista de dependencias sintácticas de los tokens (*deps*). En caso de haber coincidencia se guarda el índice del token correspondiente en la lista *advcl_detected*.
- La etiqueta “ROOT” en la lista de dependencias sintácticas de los tokens (*deps*). En caso de haber coincidencia se guarda el índice del token correspondiente en la lista *root_detected*.
- La etiqueta “Mood=Ind” en la lista de información morfológica de los tokens (*morph*). En caso de haber coincidencia se guarda el índice del token correspondiente en la lista *ind_detected*.

Al final de la iteración sobre la lista de tokens se comprueba que todas las listas tengan al menos un elemento, en caso contrario no existiría subordinación causal como se decía en su definición. Y por último si se cumple esta condición se itera sobre una lista de pares de elementos de las listas *advcl_detected* y *root_detected* creando una lista de tuplas con un valor de cada lista en cada una de ellas mediante la función zip [20]. En cada iteración de este bucle se comprueba si el primer valor de la tupla (correspondiente a la lista *advcl_detected*) es menor al segundo (correspondiente a la lista *root_detected*), es decir, se comprueba si se ha detectado una cláusula adverbial antes que un

verbo raíz, lo que implica una subordinación que cumple con la estructura sintáctica de las subordinadas causales que se quieren detectar. Si esta última condición se cumple el flag *advcl_before_root* toma el valor "True".

```
if advcl_before_root:

    startIndex = text.find(sentence.text)
    endIndex = startIndex + len(sentence.text)

    issues.append({
        "guideline": "guideline8Orac",
        "startIndex": startIndex,
        "endIndex": endIndex,
        "problematicClause": text[startIndex:endIndex],
        "explanation": "Causal adverbial subordinate clause detected"
    })
```

10 Construcción de una entrada del parámetro de salida

En caso de que no se cumpla alguna de las condiciones descritas en el flujo principal de ejecución el flag *advcl_before_root* seguiría teniendo el valor "False" y por tanto no se ejecutaría el fragmento de código de la figura *10*. Se comprueba el valor del flag de detección, si se encontró subordinación causal (valor "True") se construye una entrada de la lista devuelta como parámetro de salida. Cada entrada es un diccionario con las claves que utilizan los servicios del proyecto para las incidencias: un identificador de la pauta, el índice de comienzo de la cláusula problemática, el índice de finalización, el texto de la cláusula y una breve explicación de la causa de la incidencia.

Al final de la ejecución del bucle principal de la función que iteraba sobre las sentencias del objeto *document* la función retorna el parámetro de salida

```
return issues
```

11 Retorno de la función

Cabe destacar algunas limitaciones del modelo, esta función detecta oraciones subordinadas causales que comienzan por la palabra "como" y que establecen una relación causa-efecto pero cabe la posibilidad de que haya estructuras aún más complejas en el lenguaje en las que exista subordinación causal pero no cumplan con la estructura descrita en este trabajo. Esto podría ocurrir en casos en los que existan por ejemplo varias subordinadas encadenadas cuya estructura es mucho más compleja y difícil de detectar automáticamente que no sigan el patrón que se busca, por tanto sería un aspecto interesante a tener en cuenta para trabajos futuros. Al mismo tiempo existen otras estructuras en el español que cumplen la estructura que se ha descrito pero no son casos de subordinación causal, por ejemplo:

La oración "Como ayer estaba lloviendo, Juan cogió el paraguas" y la oración "Como vimos en el libro, no hay indicios de dinosaurios" cumplen ambas con la estructura de las oraciones que queremos detectar, sin embargo la primera es una subordinada adverbial causal y la segunda una subordinada adverbial de modo, por tanto en la primera se puede establecer una relación causa-efecto (Ayer estaba lloviendo – Juan cogió el paraguas) y en la segunda no existe esta relación de causa.

Este tipo de oraciones al tener la misma estructura sintáctica que las causales se detectarían como positivos, en este caso, falsos positivos en la función

identify. No se ha encontrado ninguna forma de diferenciarlas de las subordinadas causales con el etiquetado Part-of-Speech de spaCy durante el desarrollo del trabajo, por tanto para diferenciarlas necesitamos otra estrategia que intentaremos en la fase de pruebas.

3.2 Fase de adaptación

En la segunda y última fase de la pauta, que es la adaptación (implementada en adapt.py), se tratará de transformar una estructura que se ha detectado como subordinada causal por la función de identificación. La tarea que debe realizar la función de esta segunda fase es la de separar causa y efecto en la oración detectada por la fase de identificación y transformarlas en una sentencia compuesta por dos oraciones simples (la causa y el efecto) separadas por un punto. Para que no se pierda la relación de causalidad que había en la oración original, causa y efecto estarán unidas por un nexo causal (por eso, por ello, por tanto, así es que) tras el punto que separa ambas ideas.

Es importante mencionar una limitación que existe a la hora de adaptar las oraciones detectadas en la primera fase: la estructura formal de las subordinadas causales externas se compone del nexo "como" precediendo a la oración, una frase que identificamos como causa, una coma, y la oración principal que identificamos como efecto. Es decir algo así "Como <causa>, <efecto>". Esta es la estructura correcta, la que especifican las reglas del idioma, pero existe la posibilidad de encontrar un texto donde aparezcan estructuras subordinadas causales que no estén escritos de la manera adecuada que marcan los cánones de la lengua, como por ejemplo en el caso de que no hubiera una coma separando la causa y el efecto. Si ocurre esto, la función de identificación no será capaz de separar causa y efecto en la oración original y por tanto no podrá realizar la transformación.

La implementación funciona para las subordinadas causales que respeten la estructura que deben tener [21][22], en caso de no encontrarse una coma separando ambas ideas la transformación no funcionaría y, aunque esto no debería ser un problema, ya que las oraciones que se quieran transformar deberían ser sintácticamente correctas para poder aplicar de forma consistente las recomendaciones de Lectura Fácil, estaría bien tener este aspecto en cuenta para revisarlo en trabajos futuros y buscar una manera de transformar estas estructuras en casos en los que no exista una coma separando causa y efecto. Actualmente no se ha encontrado ninguna manera de diferenciar causa y efecto en la oración inicial mediante otros patrones sintácticos diferentes al mencionado de la separación mediante una coma.

La función de adaptación recibe cuatro parámetros de entrada:

- text -> La cadena de texto a analizar
- format_information -> Un parámetro opcional que da información adicional sobre el texto
- sentence -> El objeto sentence de spaCy (similar a document pero contiene una única sentencia)
- guideline -> Un objeto con campos que ayudan a localizar la incidencia (y que se devolverá modificado si procede en el retorno de la función)

```
def adapt(text, format_information, sentence, guideline):
```

12 Firma o signature de la función adapt

Al comienzo se inicializan los parámetros de salida de la función con unos valores preliminares que serán devueltos en caso de que no se consiga adaptar la estructura recibida.

```
simplified_text = sentence.text_with_ws
guideline['transformation'] = simplified_text
```

13 Instanciación inicial de los parámetros de salida

Tras esto se inicializan las estructuras de datos de control del programa, estas son: una lista con todos los tokens de la sentencia (*tokens*) y una lista con los valores de la variable lemma de cada token (*lemmas*). Estas estructuras de datos serán utilizadas para ejecutar la lógica del programa.

```
tokens = [t for t in sentence]
lemmas = [t.lemma_ for t in sentence]
```

14 Estructuras de datos del programa

A continuación se comprueba que la sentencia a analizar no sea una cadena vacía, ya que spaCy podría reconocer como oraciones algunas cadenas compuestas únicamente por los llamados "whitespace characters", es decir, saltos de línea, espacios en blanco, etc. En caso de que se introdujese una cadena de este tipo como parámetro de entrada, la función de adaptación debe retornar inmediatamente con los valores instanciados al inicio de la función en los parámetros de salida. Por otro lado, si la sentencia es correcta, se inicializan algunas variables importantes: un número entero, que representa el índice del token correspondiente a la coma que separa la causa del efecto en la oración original (recordemos que en teoría, las oraciones que se pasan como entrada a esta función deben ser subordinadas causales, ya que cumplen con la estructura que se busca en la fase de identificación); otro número entero que representa el índice de inicio de la cláusula problemática (en este caso siempre es cero, esto se debe a que al ser subordinadas causales, si existe una cláusula problemática, esta es la oración completa, no una parte de ella); un número entero más que representa el índice de finalización de la cláusula problemática en el texto (será el final de la oración por la misma lógica aplicada al anterior caso); y una cadena de texto que representa la cláusula problemática.

```
if len(sentence.text.strip()) > 0:
    punct_index = lemmas.index(",")
    startIndex = 0
    endIndex = startIndex + len(simplified_text)
    problematicClause = simplified_text[startIndex:endIndex]
else:
    return simplified_text, guideline
```

15 Comprobación de cadena no vacía e inicialización de parámetros

Si la sentencia no era vacía el programa continúa y se inicializan las últimas estructuras de datos necesarias para realizar la transformación, estas son: una lista con todos los signos de puntuación que pueden aparecer al final de una subordinada causal (más adelante se explica su uso); un número entero que representa el índice en el que se encuentra el token correspondiente al nexo "como" que precede a la oración; una lista con el texto de los tokens correspondientes a la causa en la subordinación causal (desde el siguiente token al asociado a la palabra "como" hasta el token correspondiente a la coma que separa causa de efecto, no incluido este último); una lista con el texto de los tokens correspondientes al efecto (desde el token asociado a la coma hasta el último token de la sentencia); y una lista con el texto de todos los tokens

incluidos en las listas de causa y efecto (todos los tokens de la oración inicial que se mantendrán en la transformación).

```
punct = [".", ",", ":", ";", "!", "?"]
conj_index = lemmas.index("como")
causa = [t.text for t in tokens[conj_index + 1:punct_index]]
efecto = [t.text for t in tokens[punct_index + 1:]]
texts = [t.text for t in tokens[conj_index + 1:punct_index]]
texts.extend([t.text for t in tokens[punct_index + 1:]])
```

16 Variables auxiliares

Ahora que se han podido aislar causa y efecto, es el momento de unirlos de la forma que especifica la pauta de Lectura Fácil. Relacionar ambos conceptos separados mediante un punto y un nexo de causalidad.

Se guarda en una variable la primera letra de la primera palabra de la que será la nueva oración, la primera de la causa (ahora veremos por qué). Se crea una lista con todos los nexos que se han considerado apropiados para unir dos ideas respetando la causalidad y la pauta de Lectura Fácil y se elige uno aleatoriamente guardando su valor en otra variable.

Ahora es el momento de modificar el valor del parámetro de salida *simplified_text* (que guardará el valor de la transformación) con lo siguiente: los valores de la lista *causa* unidos mediante espacios, seguido de la secuencia “. <nexo>, ” y por último los valores de la lista *efecto* unidos mediante espacios. El valor de nexo no es único, existen varios nexos que sirven para expresar causalidad y que pueden ser utilizados en la sustitución.

Después reemplazamos el valor de la primera letra de la primera palabra por su versión en mayúscula (ya que es el comienzo de la nueva oración) y por último iteramos sobre la lista de tokens de la nueva oración con el fin de eliminar los espacios precedentes a los signos de puntuación que aparecen en la oración transformada. Estos espacios aparecen como producto de la transformación en la que unimos causa y efecto mediante espacio, sin embargo, como sabemos, no es correcto. No puede haber espacios entre los signos de puntuación de cierre y las palabras que los preceden. Por ejemplo: las oraciones “¡Hola a todos!” y “Buenas tardes.” son correctas, sin embargo las oraciones “¡ Hola a todos ¡” y “Buenas tardes .” no lo serían.

```
first = texts[0][0]
causal_marks = ["Por eso", "Por ello", "Por tanto", "Así es que"]
nexo = random.choice(causal_marks)
simplified_text = (" ".join(causa) + f". {nexo}, " + " ".join(efecto))
simplified_text = simplified_text.replace(first, first.upper(), 1)

for word in texts:
    if word in punct:
        simplified_text = simplified_text.replace(f" {word}", word)
```

17 Transformación de la oración inicial

Para finalizar se instancia la variable *guideline* con los valores correspondientes al identificador de la pauta, el inicio, final y valor de la cláusula problemática, una breve explicación de la incidencia y el valor de la oración transformada.

Una vez hecho esto la función retorna los parámetros de salida: el texto modificado y la variable que se acaba de instanciar.

```
guideline = {
    "guideline": "guideline8Orac",
    "startIndex": startIndex,
    "endIndex": endIndex,
    "explanation": "Causal adverbial subordinate clause detected",
    "problematicClause": problematicClause,
    "transformation": simplified_text
}

return simplified_text, guideline
```

18 Instanciación y retorno de los parámetros de salida

3.3 Pruebas

En la fase de pruebas comprobaremos la fiabilidad del modelo a la hora de identificar y transformar oraciones subordinadas causales. Para ello usaremos un set de 125 oraciones descargado del corpus de la RAE [24], para descargarlo se han utilizado los criterios lema=”como” y ordenación “lema pivote”. De esta manera obtenemos un set con que contienen la palabra “como” y tas aplicar algunas pequeñas transformaciones al fichero de descarga para eliminar información descartable y eliminar duplicados pasamos a las pruebas.

Antes de empezar con las pruebas automáticas hay que comentar algunas cosas acerca del formato de representación de la información que se ha utilizado. En primer lugar, se hizo una clasificación manual de cada oración para determinar si se trataba de una subordinada causal o no. Después se incorporaron en un archivo Excel las oraciones y la clasificación de cada una (causal y no causal). Este es el fichero de origen que toman las pruebas.

Una vez que tenemos un fichero que poder leer y manipular mediante código se puede empezar a verificar el funcionamiento del proyecto.

Todas las oraciones del set se analizan con la función identify y se crea una lista que guardará el resultado devuelto (si se ha encontrado alguna incidencia en el texto guardamos el valor “causal” y en caso contrario, “no_causal”) con el fin de añadirla como nueva columna a un nuevo archivo Excel. Los valores que se guardan en esta columna son los que nos determina la función identify, pero no tienen por qué ser los correctos. Es por eso que también se crea otra lista que en cada iteración comprueba si el valor devuelto por la función identify y el valor que habíamos determinado manualmente coinciden. Esto nos permite saber si la función ha dado el resultado correcto (positivo o negativo) o si ha habido errores (falsos positivos o falsos negativos).

Los datos de las columnas originales más los de estas dos nuevas columnas se guardan en un nuevo fichero de Excel para realizar otras comprobaciones sobre ellas.

Una vez realizadas estas pruebas podemos ver algunas estadísticas acerca de la función identify:

En el set original había únicamente dos oraciones que fueran realmente subordinadas casuales, ambas han sido identificadas correctamente por la función identify (positivos). Sin embargo, hay otras once oraciones que la función ha detectado como positivos pero no lo son (falsos positivos), ¿por qué ocurre esto?

Verificando las oraciones que han sido detectadas como falsos positivos se puede ver que todas ellas son subordinadas adverbiales de modo o manera (las que comentábamos en el capítulo 3.1 que coincidían en estructura con las causales que buscamos), por tanto es algo que se sabía que podía pasar pero que ahora podemos confirmar. Por otra parte teniendo en cuenta las características del set parece que las subordinadas adverbiales de modo son bastante más frecuentes en nuestro lenguaje que las causales por tanto esto es algo que sería conveniente revisar en trabajos futuros ya que puede llevar a muchos falsos positivos.

Por último, hay que destacar que no se ha producido ningún falso negativo, es decir, que ninguna oración que fuese causal ha sido detectada como no causal.

Tras las pruebas relacionadas con la función identify pasamos a las pruebas de la fase de adaptación:

Es importante recordar que la adaptación se realiza para las oraciones que han sido detectadas como positivos en la fase de identificación, en este caso, los dos positivos y los once falsos positivos.

Para empezar vamos a separar los positivos de los falsos positivos para evaluarlos por separado. En primer lugar se crea un nuevo fichero Excel y se guardan en dos hojas separadas los resultados de las pruebas de esta fase para los positivos y para los falsos positivos.

Las pruebas para ambos casos van a ser idénticas pero se separan por comodidad. Lo primero que haremos será crear dos columnas, en una guardaremos el valor de la cláusula problemática detectada por la función identify en cada caso y en la otra el valor de la transformación de dicha cláusula realizada por la función adapt.

Por último y una vez finalizada la tarea anterior, se compara la distancia semántica que existe entre la oración original y su transformación (valor entre 0 y 1) para cada caso mediante el procesador de lenguaje natural de Hugging Face [25] y se añaden dichos resultados como una nueva columna al fichero Excel de las transformaciones.

Con estos resultados podemos determinar el buen o mal funcionamiento de la función de adaptación.

Como se comentaba más arriba, el valor devuelto por la API de Hugging Face representa la distancia semántica entre dos oraciones, es decir, cuán similar es su significado, expresado como un valor entre 0 y 1.

Cabría esperar que los resultados para los positivos fueran considerablemente mejores debido a que la transformación es más fiel, sin embargo los resultados para los fasos positivos se mueven en el mismo rango que para los positivos reales. Esto puede deberse a que los falsos positivos son todos ellos, casos de subordinación adverbial de modo y tienen una estructura muy similar a las causales. Pero también puede deberse, y es importante destacarlo, a que en los casos de los falsos positivos encontramos oraciones muy largas y con una estructura compleja en las que la transformación altera muy ligeramente su estructura. Esto a ojos de un humano es notable puesto que no se establece la relación causa efecto que debería en la transformación (ya que no son subordinadas causales realmente) pero puede que pase por alto para un procesador de lenguaje natural.

Similarity
0,983
0,961255

19 Similaridad Positivos

Similarity
0,956761
0,967675
0,990745
0,982423
0,993219
0,964348
0,965261
0,96322
0,977358
0,962303
0,945316

20 Similaridad Falsos Positivos

4 Resultados y conclusiones

En este capítulo se comentan las conclusiones extraídas durante el desarrollo del trabajo así como una pequeña reflexión personal sobre lo que ha significado el proyecto.

En cuanto a los resultados del proyecto se ha obtenido un avance en la línea que buscaba el TFG, se ha conseguido desarrollar una pauta que reconozca las subordinadas causales y en principio parece ser fiable ante falsos negativos ya que no hemos obtenido ninguno (por supuesto esto podría ser revisado con un set de oraciones mayor y quizás alguna otra técnica de testing), por tanto se ha conseguido una herramienta que puede ser útil y, aunque existe un problema de ambigüedad en cuanto a los falsos positivos relacionado con las subordinadas adverbiales de modo, es un primer paso positivo para el desarrollo de esta pauta. Si se consigue atajar este problema y los comentados en el capítulo 3 (que probablemente dependan de una mayor investigación para aumentar el conocimiento sobre este tipo de oraciones y atajar el problema de otro modo) puede ser una herramienta muy útil y muy completa para este tipo de oraciones.

Por último me gustaría comentar lo que ha significado para mí este proyecto, cuando lo escogí no sabía lo que era la metodología de Lectura Fácil ni había participado en ningún proyecto con la accesibilidad. Desarrollar este TFG ha hecho que tenga que informarme sobre un problema a solucionar muy distinto a cualquier otro que hubiese afrontado con antelación y me ha hecho plantearme la importancia de la accesibilidad en todos los ámbitos, incluido uno tan importante como es la lectura. También he aprendido tecnologías nuevas y he adquirido conocimientos sobre un área que ignoraba hasta el momento.

Por tanto considero a nivel personal muy beneficioso el desarrollo de este trabajo dado el carácter de investigación que contiene y creo que es importante que continúen haciéndose trabajos y tesis relacionados con esta línea de investigación ya que, al margen de lo que signifique para los alumnos, aporta a una buena causa y casa perfectamente con la filosofía de la carrera.

5 Trabajos Futuros

Como se ha comentado en capítulos anteriores, el modelo tiene algunas limitaciones tanto en la fase de identificación como en la fase de adaptación, por tanto existen algunas líneas abiertas que sería interesante afrontar en futuros trabajos y tesis para aumentar la flexibilidad a la hora de detectar y adaptar estar estructuras. Vamos a comentar algunas en función de la fase a la que pertenecen.

Fase de identificación:

Existen muchos tipos de estructuras en el español, algunas de ellas muy complejas y esto puede dificultar la detección de incidencias tanto por esta como por otras pautas en el proyecto. En este caso, la función de identificación podría devolver un falso negativo para oraciones subordinadas causales que tengan una estructura muy compleja que no cuadre con la propuesta en este trabajo.

Esto puede ocurrir en casos donde existan varias subordinadas encadenadas, que además podrían ser de varios tipos, lo cual haría aún más compleja la estructura y por tanto más difícil de automatizar. Actualmente no existe ninguna manera de detectar estos casos en lo que se encadenen varias subordinadas. Estudiar si existe alguna manera de distinguir e identificar estos casos en base a su estructura sintáctica o si se podría solventar mediante la ejecución de varias pautas ordenadas por una jerarquía concreta es un objetivo a considerar para el futuro.

Por otra parte, existen otros tipos de oraciones que coinciden con la estructura que se espera de las subordinadas causales y que por tanto, son indistinguibles de estas en función del etiquetado que realiza spaCy (como ya se comentaba en el capítulo 3.1). Podemos intentar detectar falsos positivos para descartar dichos resultados mediante el uso de otros procesadores de lenguaje (como se hace en el apartado 3.3) pero si se consiguiera encontrar una forma de diferenciarlas de las subordinadas causales basándose en el etiquetado sintáctico sería una mejora interesante a incorporar a la pauta.

Fase de adaptación:

En el capítulo dedicado a la fase de adaptación (3.2) se comentaba la limitación que existe en este método. Actualmente sería capaz de adaptar correctamente las subordinadas causales que cumplan con la estructura formal que deben tener (Como <causa>, <efecto>), pero en caso de encontrarse un texto en el que aparezcan este tipo de oraciones sin la coma que separa causa de efecto, la función de identificación no podría separar causa y efecto y por tanto, no realizaría ninguna transformación.

Aunque dichas estructuras no serían sintácticamente correctas en caso de no tener la coma (y por tanto en principio no debería ser motivo de preocupación), sería interesante tener un mecanismo para poder solventar este obstáculo, de manera que la función sea algo más flexible para oraciones que puedan aparecer, por ejemplo, en textos informales con oraciones parcialmente incorrectas en su estructura.

6 Análisis de Impacto

En este capítulo se destacarán también aquellas decisiones tomadas a lo largo del trabajo que tienen como base la consideración del impacto.

En este capítulo se analiza el potencial impacto vinculado a la realización de este TFG. Primero expondremos cómo este impacto se relaciona con los Objetivos de Desarrollo Sostenido referenciados en la agenda 2030 [23] y después se hará un análisis del impacto del proyecto en diferentes contextos.

Los Objetivos de Desarrollo Sostenible (ODS) son una serie de 17 objetivos definidos por la Organización de Naciones Unidas (ONU) ser un "plan para logar un futuro mejor y más sostenible para todos" según su propia definición.

El objetivo de este trabajo, al igual que todos los relacionados con la metodología de Lectura Fácil, es crear textos más accesibles que permitan igualar las oportunidades de acceso a los mismos independientemente de las capacidades del lector. Por tanto, podríamos vincularlo con el objetivo 4 (Educación de Calidad) debido a las posibilidades que aporta este proyecto y sus predecesores a la hora de adaptar libros de texto o literarios para alumnos con dificultades en la comprensión lectora, democratizando así el acceso a la educación y la cultura.

De la misma manera podemos vincular el impacto del proyecto con el objetivo número 10 (Reducción de las Desigualdades) puesto que la incorporación de alternativas accesibles de los textos clásicos así como de las nuevas obras que se producen a diario y los textos académicos entre otros, permite el acceso a ciertos recursos e información a sectores poblacionales que antes no habrían podido acceder a ellos, favoreciendo de esta manera la equiparación de derechos y oportunidades de forma directa.

Fuera de estos objetivos, se puede analizar el impacto del proyecto en diferentes contextos.

Personal:

A nivel personal el desarrollo de este trabajo me ha permitido ampliar mis conocimientos en varios horizontes nuevos. El primero, aunque no es el más relevante para el proyecto, ha sido el aprendizaje del lenguaje de programación Python, del cual solo conocía conceptos muy básicos antes de la realización del proyecto. En segundo lugar, me ha permitido ampliar considerablemente mis conocimientos acerca del campo del procesamiento del lenguaje natural y la utilización de un modelo pre entrenado de ML como es spaCy para crear proyectos e investigar en esta área del conocimiento.

Por último creo que es beneficioso aprender y entender qué trata la metodología de Lectura Fácil y darse cuenta de la importancia que tiene la accesibilidad en el mundo actual en todos los ámbitos imaginables. Una de las máximas de los países modernos y los estados de derecho es la igualdad de los individuos, ya sea ante la ley, a la hora de acceder al mercado laboral o a la hora de acceder al conocimiento o la cultura como en este caso. Si consideramos que este es un objetivo justo y que se debe preservar es imprescindible que existan proyectos como este cuyo objetivo es hacer el mundo un poco más justo en su ámbito de aplicación.

Empresarial:

En cuanto al impacto que este y los otros proyectos desarrollados en esta línea de investigación pueden tener en el mundo empresarial son varios.

Estos proyectos pueden derivar en la creación de aplicaciones que sirvan como herramienta de trabajo para aquellos profesionales y/o empresas que se dedican a adaptar textos en base a las recomendaciones de Lectura Fácil. También pueden servir para evaluar si un texto que ya fue adaptado manualmente tiene alguna incidencia no resuelta o que se ha pasado por alto, para adaptar textos y documentos internos de una empresa a empleados que tengan alguna discapacidad o trastorno del lenguaje, o para adaptar grandes obras de la literatura que tienen un vocabulario y estructuras complejas para lectores más jóvenes o con problemas de comprensión lectora, entre otras.

Social:

El impacto de este trabajo en el contexto social tiene mucho que ver con las relaciones descritas entre el proyecto y los Objetivos de Desarrollo Sostenible. Por evita redundancias podríamos resumirlo en la capacidad que este y el resto de proyectos relacionados con Lectura Fácil tienen para facilitar el acceso a la cultura y el conocimiento escrito para los sectores de la población hacia los que está dirigida la metodología, así como el fomento de oportunidades y la reducción de la desigualdad que genera que toda esa gente tenga acceso, al igual que el resto, a dichos recursos.

Cultural:

De nuevo al igual que en el contexto social, el impacto cultural del proyecto está muy ligado a los ODS por la capacidad de equiparación de oportunidades que supone. Los textos accesibles proporcionan un mayor grado de inclusión de la población con problemas de comprensión lectora en todos los ámbitos de su vida.

7 Bibliografía

[1] lectura-facil-metodos.pdf (plenainclusion.org)

[2] https://spacy.io/

[3] TFG_GUILLERMO_GIL_DE_EUSEBIO1.pdf (upm.es)

[4] TFG_ALEJANDRO_MING_DIAZ1.pdf (upm.es)

[5] TFG_JULIA_ANTONA_PALACIOS1.pdf (upm.es)

[6] Andrew Ng, Chief Scientist at Baidu - YouTube

[7] ¿Qué es una Red Neuronal? Parte 1 : La Neurona | DotCSV - YouTube

[8] ¿Qué es el Procesamiento de Lenguaje Natural y cómo aplicarlo? - YouTube

[9] Linguistic Features · spaCy Usage Documentation

[10] Primeros pasos en NLP con spaCy, un vistazo general. | by Gerardo Fosado | Medium

[11] Annotation Specifications · spaCy API Documentation

[12] Library Architecture · spaCy API Documentation

[13] Las oraciones subordinadas adverbiales causales (sintaxis.org)

[14] ORACIONES CAUSALES. Porque, como, ya que, puesto que... - Divinas Palabras. Victoria Monera

[15] UNE 153101:2018 EX Lectura Fácil. Pautas y recomendaciones par...

[16] Ley 6/2022, de 31 de marzo, de modificación del Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social, aprobado por el Real Decreto Legislativo 1/2013, de 29 de noviembre, para establecer y regular la accesibilidad cognitiva y sus condiciones de exigencia y aplicación para vuestro conocimiento. - Accessibilitas

[17] Install spaCy · spaCy Usage Documentation

[18] Rule-based matching · spaCy Usage Documentation

[19] spaCy - Doc.sents Property (tutorialspoint.com)

[20] Python zip() Function (w3schools.com)

[21] 662072.pdf (uvigo.gal)

[22] Gramática española: Oraciones subordinadas causales: clases, nexos y ejemplos (elblogdegramatica.blogspot.com)

[23] Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible (un.org)

[24] CREA | Real Academia Española (rae.es)

[25] What is Sentence Similarity? - Hugging Face

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Sun Jan 15 22:38:43 CET 2023
	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)