Recolección de datos para el generador de historias con género literario a partir de imágenes utilizando "*Computer Vision*" y *Natural Language Processing*

Oscar Andres Ramos Maldonado

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

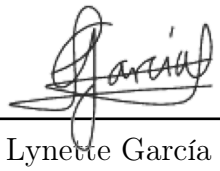
UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Recolección de datos para el generador de historias con género literario a partir de imágenes utilizando "*Computer Vision*" y *Natural Language Processing*

Trabajo de graduación en modalidad de tesis presentado por
Oscar Andres Ramos Maldonado
Para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala, 5 de diciembre del 2023

Vo.Bo.:

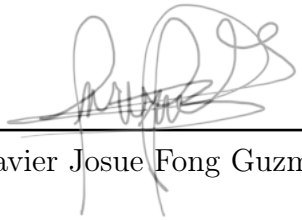
(f) 

Lynette García Perez

Tribunal Examinador:

(f) 

Lynette García Perez

(f) 

Javier Josue Fong Guzman

(f) 

Eddy Omar Castro Jauregui

Fecha de aprobación: Guatemala, 5 de diciembre de 2023.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo de graduación se enfoca en la generación de una metodología que facilite la recolección y procesamiento de datos destinados a entrenar un *large language model* (LLM) para garantizar la calidad y eficacia del modelo resultante y para optimizar el proceso de entrenamiento.

Se dio un enfoque principalmente a la recolección inicial de archivos PDF utilizando un *web scraper*. Con esto se propone un marco de trabajo para la limpieza de datos que abarca desde la detección y eliminación de valores atípicos hasta la generación de reportes sobre las tendencias encontradas en los archivos, lo que mejora su coherencia y cohesión.

Por otro lado, se evaluó la calidad de los datos mediante el entrenamiento de un modelo *GPT-2 (generative pre-trained transformer)* de pequeña escala y comparando los textos generados por el modelo con los datos limpios y el texto extraído directamente de los archivos.

Este proyecto contribuye al avance en inteligencia artificial y el procesamiento del lenguaje natural al establecer tanto pautas como buenas prácticas para la recolección y limpieza de datos destinados a entrenar modelos de lenguaje.

CAPÍTULO 1

Introducción

Durante el último año, el uso de *LLM* se ha popularizado debido a la alta disponibilidad de los modelos y sus varias aplicaciones. A pesar de que la calidad de los datos de entrenamiento forma una parte integral en la calidad del producto final, no se tiene una guía concreta sobre cómo se deben procesar los datos para garantizar un resultado de calidad. Esto se debe, principalmente a que el tipo de procesamiento dependerá del requerimiento de los datos, el contexto y lo que se quiere lograr con el modelo resultante.

Este trabajo de graduación se centra en el desarrollo de una metodología general para la recolección y la limpieza de archivos PDF que contienen historias literarias en español de diferentes géneros para entrenar un *LLM* con el fin de ilustrar el proceso de recolección y limpieza de datos. La correcta preparación de los datos no solo garantiza la calidad del modelo resultante, sino que también optimiza el proceso de entrenamiento, reduciendo tiempo y recursos necesarios. Se recolectaron de archivos PDF desde páginas web con el fin de proponer una metodología de preparación de datos.

La necesidad de esta metodología surge de la carencia de pautas establecidas para la preparación de datos específicos para los *LLM*, que puede resultar en modelos sesgados o ineficientes. Al estandarizar y mejorar la calidad de datos de entrenamiento se busca contribuir al avance de la construcción de modelos de lenguaje más precisos y versátiles.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Definir una metodología que permita producir un conjunto de datos limpios y preparados para el entrenamiento de modelos generativos de lenguaje en idioma español.

2.2. Objetivos específicos

- Seleccionar datos para formar un conjunto que permita el entrenamiento de modelos generativos de lenguaje en idioma español.
- Realizar un análisis exploratorio de los datos para comprender mejor su estructura, características y patrones.
- Realizar procesos de limpieza de datos que permitan aumentar la calidad de los datos.
- Entrenar un modelo de lenguaje que permita probar la metodología propuesta.

CAPÍTULO 3

Justificación

Este proyecto ejemplifica la recolección y limpieza de datos destinados a entrenar un *LLM*. Los datos de entrenamiento son un recurso, por lo que su correcta utilización puede marcar una gran diferencia en el desempeño del modelo.

En la era actual de la información, la cantidad de datos disponibles se ha multiplicado exponencialmente, lo que resulta en presentaciones de información de varios tipos (Duarte, 2023). La correcta preparación y limpieza de estos datos es esencial para garantizar que un modelo entrenado refleje resultados coherentes; por lo que una metodología bien definida permite abordar desafíos como la heterogeneidad de los datos, presencia de ruido y errores, sesgos lingüísticos, etc.

La metodología propuesta busca contribuir a la mejora y disponibilidad de idiomas de los *LLMs* al proporcionar un ejemplo claro, así como técnicas eficientes para la recolección y limpieza de datos.

CAPÍTULO 4

Marco Teórico

4.1. Large language models *(LLM)*

Los modelos de lenguaje son un tipo de aprendizaje de máquinas, capaces de una gran variedad de tareas de procesamiento de lenguaje natural (NLP por sus siglas en inglés), por ejemplo: generación de texto, traducciones, clasificaciones y respuesta a preguntas de forma conversacional. Cuentan con una gran cantidad de parámetros, de tal forma que los más exitosos controlan miles de millones. Para el efecto, se les entrena mediante aprendizaje autosupervisado para predecir tokens en una oración según el contexto. El proceso se repite hasta que el modelo alcanza el nivel deseado de precisión. Una vez se haya entrenado un modelo, puede ser refinarse para cumplir varias tareas como responder preguntas con base en documentos, analizar la retroalimentación de clientes, clasificar textos, etc. (Barney, 2023).

Los LLM utilizan redes neuronales para generar productos basados en la información aprendida durante el entrenamiento. Actualmente las redes de los modelos tienden a estar formadas por arquitecturas de transformadores, las cuales utilizan una técnica llamada auto atención que les permite prestar atención a diferentes partes de la secuencia de entrada para hacer predicciones. Primero, calculan una suma de pesos de la entrada para determinar, dinámicamente la relevancia de cada token con cada token. Luego, la relevancia se calcula con base en puntuaciones de atención que representan la importancia de cada token según el contexto de la secuencia (Rouse, 2023).

4.2. Tokens

En los modelos de lenguaje, un token es la representación mínima de texto, (varias palabras, una palabra, parte de una palabra o un solo caracter).Se utilizan para que la computadora procese y entienda el lenguaje. Además, sirven utilizan como entrada y salida del modelo, y durante el entrenamiento para que el modelo aprenda a predecir los siguientes tokens basándose en los previos (Maeda, 2023).

4.3. *(Generative pre-trained transformer 2)* (GPT-2)

Es un modelo de lenguaje que se basa en la arquitectura de transformadores. Cuenta con 1.5 mil millones de parámetros que generan texto prediciendo palabra por palabra. Este modelo es la versión anterior del famoso GPT-3, el cual cuenta con 175 mil millones de parámetros (Kumar, 2023). Sin embargo, debido a su tamaño más pequeño, es menos versátil en la generación de texto y en las posibilidades de refinamiento, pero esto permite que sea entrenado y ejecutado con menos recursos computacionales.

4.4. NanoGPT

Es un modelo de lenguaje basado en la arquitectura de GPT-2 que fue diseñado con fines didácticos por uno de los fundadores de OpenAI, Andrej Karpathy. A diferencia de otros modelos *nanoGPT*, es una versión sumamente liviana, lo que permite que se realice el proceso de pre entrenamiento y afinación en computadores con bajos recursos al costo de precisión y rendimiento del modelo (Kaparthy, 2023).

4.5. Entrenamiento de modelos

Generalmente los modelos son entrenados con grandes conjuntos de datos de temas generales para que extraigan características que puedan ser transferidas en una tarea específica. Este proceso involucra varios pasos: el preprocesamiento de los datos para convertirlos en una representación numérica que la computadora pueda entender; la asignación aleatoria de los parámetros del modelo; la alimentación de representaciones de los datos; la utilización de una función de pérdida para medir la diferencia entre las producciones; la optimización de los parámetros del entrenamiento para minimizar la pérdida; y la repetición hasta que se llegue a un nivel aceptable de precisión (Rouse, 2023).

4.6. Web scraping

El *web scraping* es el proceso de extraer información o datos de páginas web para exportarlos en un formato útil para el usuario. Aunque se puede realizar manualmente existen varias formas de automatizar el proceso por medio de scripts. El involucra seleccionar una página web, analizar su estructura para entender la estructura de los datos, ingresar a la página web, analizar sintácticamente el contenido de la página para extraer la información y, finalmente, almacenar los datos deseados. Este proceso es completamente legal si la información que se está almacenando no está protegida por ningún tipo de ley, ya sea de derechos de autor o de confidencialidad. Vale la pena mencionar que la práctica puede resultar en violaciones de términos de servicios o sobrecargas de servidores. En este tipo de situaciones, propietarios de los servicios generalmente bloquean la IP o terminan las relaciones con el usuario, pero la severidad de las consecuencias queda a discreción de los dueños del servicio (Rouse, 2023).

4.7. Modelación de temas

Es una técnica de procesamiento de lenguaje natural y aprendizaje automático cuyo objetivo es descubrir los temas latentes que puede tener un cuerpo de texto. Los temas son representados por patrones recurrentes en el contenido, por lo que cada uno está asociado con un conjunto de palabras que tienen alta probabilidad de aparecer juntas en relación con el tema. Una vez el modelo ha generado sus resultados, la definición de cada tema queda a interpretación del usuario (Pascual, 2019).

4.8. UNICODE

Es un estándar de codificación de caracteres cuyo fin es representar la mayoría de caracteres utilizados en todos los lenguajes del mundo. Asigna un número único llamado UNICODE a cada caracter único, incluye números, letras, símbolos y caracteres especiales de diferentes idiomas y escrituras (Tasker, 2021).

4.9. Similaridad de Jaccard

Es una medida común de proximidad utilizada para computar la similaridad entre dos vectores binarios asimétricos, por ejemplo: texto. El cálculo se realiza convirtiendo el texto en 2 vectores separados y dividiendo la intersección de los mismos dentro de la unión (Figura 1). El resultado de la operación es un valor entre 0 a 1, entre más cerca a 1 sea el resultado mayor similaridad entre los datos (Karabiber, 2023).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Figura 4.1: Fórmula de Similaridad de Jaccard

4.10. Distancia de Levenshtein

Es una forma de medir la diferencia entre 2 textos, la cual se calcula verificando la menor cantidad de cambios de edición necesarios para que ambos sean iguales (Figura 2). Los cambios de edición son inserción, eliminación y sustitución de caracteres (Grashchenko, 2023).

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figura 4.2: Fórmula de Distancia de Levenshtein

4.11. Similaridad de Coseno

Es una forma de medir qué tan cercanos son 2 cuerpos de texto en términos de contexto y significado sin importar su tamaño, ideal para textos grandes. En este caso las palabras son representadas como un vector y los documentos, como espacios vectoriales de n-dimensiones. Desde un punto de vista matemático, se mide el ángulo entre 2 vectores n-dimensionales proyectados en un espacio multidimensional para devolver un valor entre 0 y 1 en donde la cercanía a 1 indica el nivel de similitud de ambos textos (Alake, 2023).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figura 4.3: Fórmula de similaridad de Coseno

4.12. Expresiones regulares

Es una colección de caracteres que permiten crear patrones que ayudan a comparar, encontrar y manejar texto en un ambiente digital. Están compuestas de caracteres, metacaracteres, anclas, cuantificadores, clases de caracteres, secuencias de escape y modificadores (Figura 4). Los caracteres se refieren a los caracteres de texto que son tratados como tales. Los metacaracteres tienen significados especiales que permiten realizar operaciones con las expresiones. Por otro lado las anclas son metacaracteres que indican el inicio y final de una lista de caracteres. Las clases de caracteres permiten definir conjuntos de caracteres. Las secuencias de escape son combinaciones de caracteres que representan un caracter especial o una función de control, como la tabulación. Por último, los modificadores son caracteres especiales que modifican el comportamiento de las expresiones regulares al realizar operaciones de comparación, búsqueda y manejo de texto (Computer Hope, 2023).

4.13. Graphics Processing Unit (GPU)

Es el componente de una computadora que se encarga de procesar y mostrar las gráficas en un monitor. Es crucial para ejecutar múltiples operaciones matemáticas complejas en paralelo, por lo que resulta especialmente efectivo para los cálculos de entrenamiento de un modelo de inteligencia artificial. Actualmente las únicas empresas que diseñan *GPU* son AMD, Intel y NVIDIA de las cuales NVIDIA es la única que ofrece una plataforma para desarrolladores puedan utilizar los recursos en operaciones de inteligencia artificial (Gillis, 2023).

4.14. Hugging Face

Es una plataforma enfocada en la inteligencia artificial y *data science* que permite crear, lanzar y entrenar modelos de *machine learning*. También proporciona infraestructura para correr y desplegar modelos en formato *open source* lo que permite compartir el trabajo para que otros usuarios aporten sus ideas (Rebelo, 2023).

Character	What does it do?	Example	Matches
^	Matches beginning of line	^abc	abc, abcdef.., abc123
$	Matches end of line	abc$	my:abc, 123abc, theabc
.	Match any character	a.c	abc, asg, a2c
|	OR operator	abc|xyz	abc or xyz
(...)	Capture anything matched	(a)b(c)	Captures 'a' and 'c'
(?:...)	Non-capturing group	(a)b(?:c)	Captures 'a' but only groups 'c'
[...]	Matches anything contained in brackets	[abc]	a,b, or c
[^...]	Matches anything not contained in brackets	[^abc]	xyz, 123, 1de
[a-z]	Matches any characters between 'a' and 'z'	[b-z]	bc, mind, xyz
{x}	The exact 'x' amount of times to match	(abc){2}	abcabc
{x,}	Match 'x' amount of times or more	(abc){2,}	abcabc, abcabcabc
{x,y}	Match between 'x' and 'y' times.	(a){2,4}	aa, aaa, aaaaa
*	Greedy match that matches everything in place of the *	ab*c	abc, abbcc, abcdc
+	Matches character before + one or more times	a+c	ac, aac, aaac,
?	Matches the character before the ? zero or one times. Also, used as a non-greedy match	ab?c	ac, abc
\	Escape the character after the backslash or create an escape sequence.	a\sc	a c

Figura 4.4: Ejemplos de notaciones en expresiones regulares.

CAPÍTULO 5

Metodología

5.1. Recolección y procesamiento de imágenes

Inicialmente se planteó la posibilidad de llevar a cabo la recolección y el procesamiento de un conjunto de personalizado imágenes para el entrenamiento del modelo de visión por computadora. No obstante, tras un análisis más profundo, se determinó que esta tarea no era necesaria ni eficiente por varias razones. Uno de los factores más influyentes fue la abundante disponibilidad de conjuntos de datos públicos ya preparados. Estos conjuntos contienen miles de imágenes que abarcan una variedad de contextos y escenarios, junto con descripciones detalladas. Esta variedad de datos permite abordar la tarea de visión por computadora de manera efectiva sin la necesidad de un esfuerzo adicional de recolección y etiquetado. Otro factor relevante fue la falta de posibilidades de automatización del proceso de creación de un conjunto de datos personalizado. Dado que la tarea se limitaba a descargar imágenes y agregar descripciones, se consideró que el trabajo manual requerido para recopilar y etiquetar un número significativo de imágenes no justificaba el tiempo y los recursos necesarios para realizarlo. Los conjuntos más relevantes para entrenar un modelo de visión de computadora son *ImageNet*, *COCO* y *MNIST*.

5.2. Web scraping

Para el manejo de archivos y reportes se utilizó el lenguaje de programación *Python* (versión 3.11.5), debido a la gran cantidad de librerías que provee para la manipulación de texto y datos. En cuanto al *web scraper*, se utilizó la librería *Beautiful Soup* por su facilidad de uso, manejo robusto de errores y sistema de análisis de texto completo.

Para desarrollar el *web scraper*, primero se necesitó una página para extraer los datos, por lo que se buscó en *Google* alguna que proveyera novelas clásicas gratis. Se optó por novelas clásicas ya que estas no están protegidas por las leyes de derechos de autor. Para elegir la página, se tomó en cuenta que tuviera una gran variedad de documentos de diferentes géneros literarios para no sesgar el conjunto de datos y que los documentos estuvieran, preferiblemente, en formato PDF para manejarlos con facilidad. Tomando en cuenta estas necesidades se analizó la página infolibros.org para determinar cómo se obtendrían los archivos.

Infolibros se encuentra dividida por categorías (amor, novelas, terror, etc.); dentro de estas, existen subcategorías como libros de amor para adolescentes, novelas clásicas, terror juvenil, etc. En estas, se encuentran los hipervínculos de descarga de los archivos PDF. Con esto, ya se diseñó el *web scraper* el procesamiento de cada una de estas páginas de subcategorías. Para agilizar el proceso, se creó un archivo TXT en el que se almacenaron las direcciones web que contienen los hipervínculos de descarga, con el fin de que el web scraper tomara cada hipervínculo y extrajera los documentos. El proceso de extracción de hipervínculos se realizó a travez de *Beautiful Soup* con un árbol de análisis sintáctico basado en la estructura de la página. Se recorrió el árbol para obtener todos los objetos tipo *href*, los cuales contienen a los hipervínculos. Para navegar la página era necesario tener referencias a otras secciones, por lo que se requirió filtrar los hipervínculos que no contienen la dirección de descarga de los documentos. Para lograr esto, se tomaron en cuenta únicamente los hipervínculos que contenían el texto *.pdf* o *drive.google.com* ya que los documentos que se encuentran guardados dentro del servidor de la página web cuentan con la frase *.pdf* o alternativamente son almacenados en una carpeta de *Google Drive*.

Una vez se tuvieron los hipervínculos, se envió una solicitud para acceder al contenido que, en este caso, eran los archivos PDF. Antes de descargarlos, se evaluó la similitud del nombre contra un listado de nombres del directorio de descargas de PDF utilizando la similaridad de Jaccard y la distancia de Levenshtein con el fin de filtrar documentos con nombres repetidos. Para garantizar una buena comparación de nombres, primero fue necesario estandarizarlos; en este caso, se creó una función que convierte todas las letras a minúsculas y elimina los caracteres que no son alfanuméricos. Una vez formateados se calcula la similaridad de Jaccard creando dos objetos tipo *set* de *Python* para ambos nombres, esto crea una lista de todos los caracteres únicos dentro del nombre. Con base en estos *sets*, se calculó la intersección y la unión para luego dividirlos según la función, lo que nos retorna un valor entre 0 y 1. Luego, para calcular la distancia de Levenshtein se utilizó la librería *Fuzzywuzzy* que cuenta con el método *ratio*, el cual realiza el cálculo al enviarle los nombres en formato de cadena de caracteres como parámetro. Esto retornó un valor entre 0 y 100 que debe ser dividido dentro de 100 para poder calcular un promedio de ambas puntuaciones. Se calculó un promedio de ambos métodos para una comparación más precisa ya que ambos cálculos toman diferentes características para la puntuación de similitud. Esta puntuación comparó con una variable que define un límite de similitud: si el valor de los cálculos era mayor a la variable límite,el nombre de ambos documentos es demasiado similar, por lo que será descartado.

Durante el desarrollo de la metodología, el valor de la variable límite fue cambiando según se observaban las tendencias de los resultados, pero, eventualmente se llegó al valor 0.55, ya que este permitió descartar la mayoría de títulos con varianzas simples como números o el orden las palabras en el nombre; al mismo tiempo, permitió que se descargaran documentos con títulos muy similares, por ejemplo: secuelas. De igual forma este valor bajo creó una situación en la que se descargaban documentos que ya se tenían pero contenían un nombre lo suficientemente diferente para superar esta barrera. Debido a que no se pudo encontrar una solución en esta etapa de la metodología, esto será abarcado más adelante. En la Tabla 5.1 se muestran algunos ejemplos de archivos que no fueron descargados por la similitud entre sus nombres.

Nombre del archivo a descargar	Nombre del archivo original	Puntuación
1 otelo autor william shakespeare	19 otelo autor william shakespeare	0.828
8 namiko autor tokutomi roka	16 namiko autor tokutomi roka	0.808
cyrano de bergerac edmond rostand	cyrano de bergerac edmond rostand	1
romeo y julieta william shakespeare	romeo y julieta william shakespeare	1
las penas del joven werther johann wolfgang von goethe	las penas del joven werther johann wolfgang von goethe	1
9 trist n e isolda autor richard wagner	14 trist n e isolda autor richard wagner	0.869
10 la dama de las camelias autor alejandro dumas	18 la dama de las camelias autor alejandro dumas	0.89
5 el castillo de otranto autor horace walpole	15 el castillo de otranto autor horace walpole	0.884
la casa en el confin de la tierra william hope hodgson	la casa en el confin de la tierra william hope hodgson	1

Tabla 5.1: Reporte de documentos descartados con su puntuación de similitud.

Otro problema fue que después de cierta cantidad de descargas a través de hipervínculos de *Google Drive* se activa una medida anti *web scraping*, resultando en el bloqueo de las próximas descargas y el retorno del código de error 403. Este problema no tiene solución debido a que la limitación de descargas por medio de *web scraping* es una decisión de términos de uso de la empresa, por lo que no se sabe cómo detectan esto o cómo realizar el proceso con su aprobación. A pesar de ello se no bloquean las descargas manuales. Además la restricción parecía desaparecer en un período de 30 a 60 minutos.

5.3. Conversión de PDF a TXT

Una vez se tuvieron todos los documentos PDF que se quieren procesar, debían convertirse a formato TXT para facilitar la manipulación del contenido durante la limpieza de los datos. En este caso, se quería tener un conjunto de datos que un *LLM* pueda procesarlo y generar historias de diferentes géneros literarios, por lo que fue necesario extraer únicamente la historia e ignorar elementos como el título, el índice, los agradecimientos, etc. En la gran mayoría, el título, el índice, los pies de página y los anuncios al final del documento se encontraban en un tipo de letra diferente a la del texto principal. Tomando esto en cuenta se realizó una función que analiza el documento PDF para obtener el tipo de letra predominante y así extraer únicamente el texto que contiene la historia.

El análisis de tipos de letra se realizó con la librería *PyMuPDF*, la cual permite extraer los metadatos de cualquier archivo PDF. Utilizando esta se recorrió cada texto, palabra por palabra, documentando las características del tipo de letra (Figura 5.1). Concluido esto se generó un reporte de los hallazgos, el cual contiene el nombre de todos los tipos de letra, su tamaño y la cantidad de palabras que la usan. Con base en esto, se ordenó el tipo de letra y su tamaño en orden descendiente para identificar cuáles eran los más predominantes y así ignorar el resto.

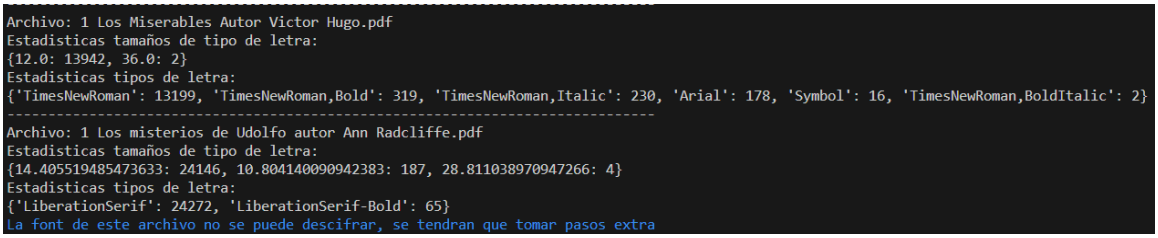


Figura 5.1: Reporte tipos de letras en archivos PDF.

Durante este proceso se notó que algunos archivos TXT en vez de tener el símbolo *UNICODE* U+0020, el cual representa el espacio que separa las palabras, tenían el símbolo U+FFFD, el cual es representado con un signo de interrogacíon dentro de un cuadrado o rombo, lo que indica que hubo un error en la interpretación del documento PDF por parte de la librería *PyMuPDF*. Observando los resultados de los reportes de tipo y tamaño de letras, se dedujo que el problema surgía en documentos cuyo tipo de letra era *Liberation Seriff*, por lo que se tuvo que añadir una función que al detectar un documento con dicho tipo de letra, realizara ciertos pasos extra para extraer el texto. Para solucionar este problema, se empleó la librería *PyPDF2*, que permite la manipulación de PDF por medio de *Python*, al igual que *PyMuPDF*. La diferencia entre ambas librerías es que *PyPDF2* tienen la capacidad de interpretar el tipo de letra *Liberation Seriff*, aunque no puede extraer los metadatos de tipos de letra dentro del documento; por lo tanto, las dos se utilizaron de forma paralela. Se utilizó *PyMuPDF* para determinar qué páginas dentro del documento tenían texto y no contaban con el tipo de letra deseado para que *PyPDF2* tomara únicamente las páginas necesarias al escribir el archivo TXT.

Otro problema que surgió fue que, dentro del formato de párrafos de algunos archivos PDF, si una oración no termina dentro del espacio de una línea, no existe un caracter que separe la última letra de la primera línea de la primera letra de la siguiente línea. Para corregir esto se colocó un espacio al inicio de cada línea, lo que solucionó el problema de separación entre palabras, pero también añadió un espacio innecesario e inconsistente al inicio de nuevos párrafos. Por razones de formato de los archivos PDF no se encontró una forma de prevenir este error en esta etapa, por lo que se abarcó durante la limpieza de datos.

Una vez superados estos problemas se ejecutó el proceso de conversión en todos los documentos PDF y se creó un directorio llamado *txt_ files* para realizar un análisis exploratorio de los resultados.

5.4. Limpieza y estandarización de datos

Con el fin de limpiar los archivos TXT, primero, se creó una copia de estos para trabajar con ellos sin volver a procesar los PDF en caso de algún un cambio indeseado o un resultado inesperado durante el proceso de limpieza. También fue necesario un análisis exploratorio para encontrar las tendencias de los datos indeseados y asegurarse de que no se borraran o editaran datos innecesarios. Por medio de este análisis, se detectaron archivos vacíos que provenían de archivos PDF con fotos de texto en vez de texto escrito. La solución fue implementar una función que detectara archivos TXT vacíos para de borrarlos junto con su contraparte en formato PDF.

Una vez se tuvieron únicamente los archivos con texto, se tomó en cuenta el idioma, pues el objetivo era entrenar un modelo de texto en español, por lo que se eliminó cualquier texto escrito en otro idioma. Esto se realizó utilizando la librería *Langdetect* y su método *detect*, el cual al enviarle un pedazo de texto, analiza la tendencia de las palabras y caracteres dentro del texto para identificar el idioma. Para que el proceso no demorara demasiado, se tomaron los primeros 2500 caracteres de cada libro, ya que, por medio del análisis exploratorio, se determinó que, aunque existía texto

indeseado al inicio, era solamente unas cuantas palabras. Gracias a esto se garantizó la detección del idioma sin tener que analizar todo el documento. En este caso se esperaba que el método retornara la abreviatura «es», que indica que el texto está en español. Si se obtenía otra archivo era borrado. En la Tabla 5.2 se muestran los nombres de los archivos con las abreviaturas de los idiomas detectados.

Nombre archivo	**Idioma**
26. Little Women (Inglés) autor Louisa May Alcott.txt	en
27. The Great Gatsby (Inglés) autor F. Scott Fitzgerald.txt	en
28. The Scarlet Letter (Inglés) autor Nathaniel Hawthorne.txt	en
29. O grande Gatsby (Portugués) autor F. Scott Fitzgerald.txt	pt
30. A Dama das Camélias (Portugués) autor Alejandro Dumas.txt	pt
31. Os Miseráveis (Portugués) autor Victor Hugo.txt	pt
13. Crime and Punishment (Inglés) autor Fyodor Dostoyevski.txt	en
15. The Black Cat (Inglés) autor Edgar Allan Poe.txt	en
17. Crime e Castigo (Portugués) autor Fiodor Dostoyevski.txt	pt
18. O estranho caso do Dr. Jekyll e Sr. Hyde (Portugués) autor Robert Louis Stevenson.txt	pt
19. O gato preto (Portugués) autor Edgar Allan Poe.txt	pt

Tabla 5.2: Reporte de archivos TXT en otros idiomas.

El siguiente paso para la limpieza fue corregir las variaciones de espacios entre palabras, identificadas en el análisis exploratorio, así como el espacio extra al inicio de los párrafos que se mencionó durante la etapa de extracción de texto. El primer problema se solucionó con la librería nativa de expresiones regulares *RE* de *Python*. Esta librería permitió sustituir los patrones de más de dos caracteres de espacios seguidos con el método *sub*, el cual recibe el patrón que se desea sustituir como expresión regular, el nuevo patrón como expresión regular y el texto a editar. La expresión regular estaba conformada únicamente por el caracter de espacio y el símbolo «+» (representación de una o más recurrencias del caracter de espacio); esto se reemplazó por un solo caracter de espacio. La solución para el inicio de párrafos fue más simple: primero, se verificó si al inicio de cada línea existía un caracter de espacio; de ser así, se cortaba el texto para ignorarlo. La Figura 5.2 muestra un archivo que contiene cuatro espacios de separación para cada palabra.

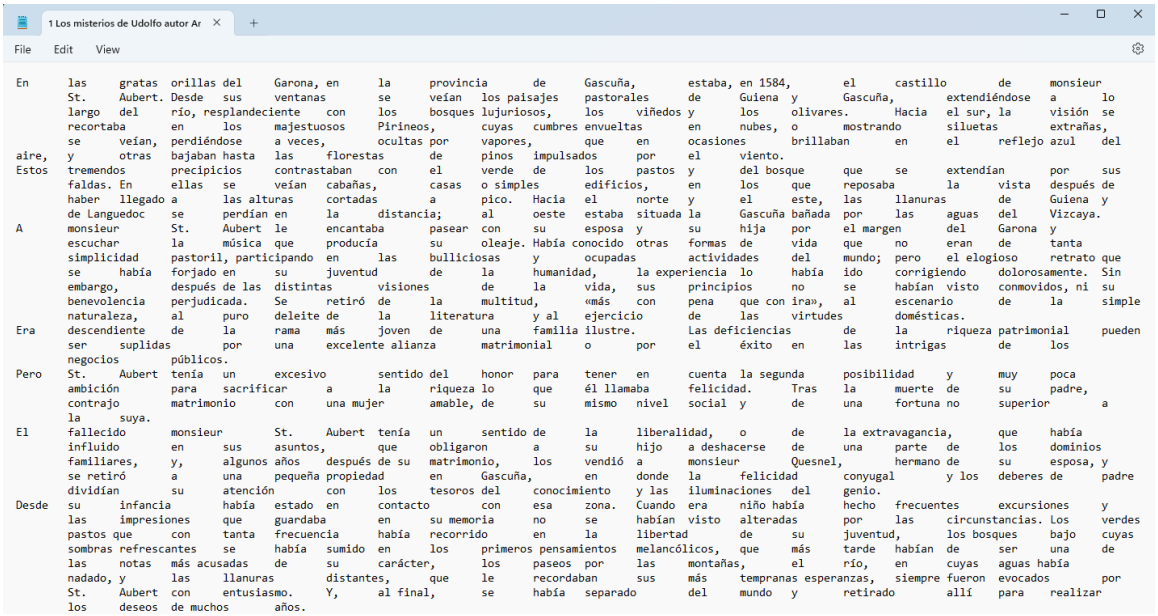


Figura 5.2: Ejemplo irregularidad de espacios.

Luego de estandarizar los caracteres de espacio, se creó un conjunto de caracteres no permitidos con el fin de eliminarlos de todos los archivos. Para lograrlo se creó una función que recorre el directorio de los archivos TXT y agrega cada caracter único a una lista (Figura 5.3).

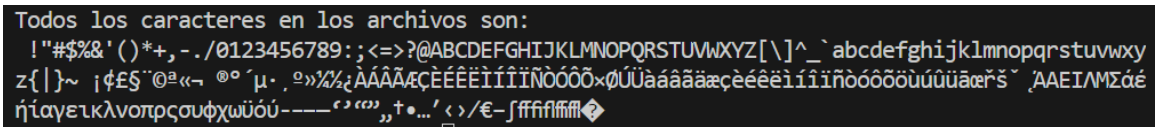


Figura 5.3: Listado de caracteres únicos en todos los archivos.

Como se puede observar, se encontraron los caracteres esperados como signos de puntuación y todo el alfabeto español, pero también caracteres de otros idiomas y símbolos variados. Tomando en cuenta que ya se había realizado un análisis de idiomas para garantizar que los textos estuvieran en español, se necesitaba contexto para estos caracteres en otros idiomas con el fin de decidir si vale la pena eliminarlos. Para esto se creó un reporte que muestra el contexto de cada uno de estos caracteres (Figura 5.4). En cambio los símbolos variados no se les dio contexto, ya que eliminarlos no cambiaba el significado del texto.

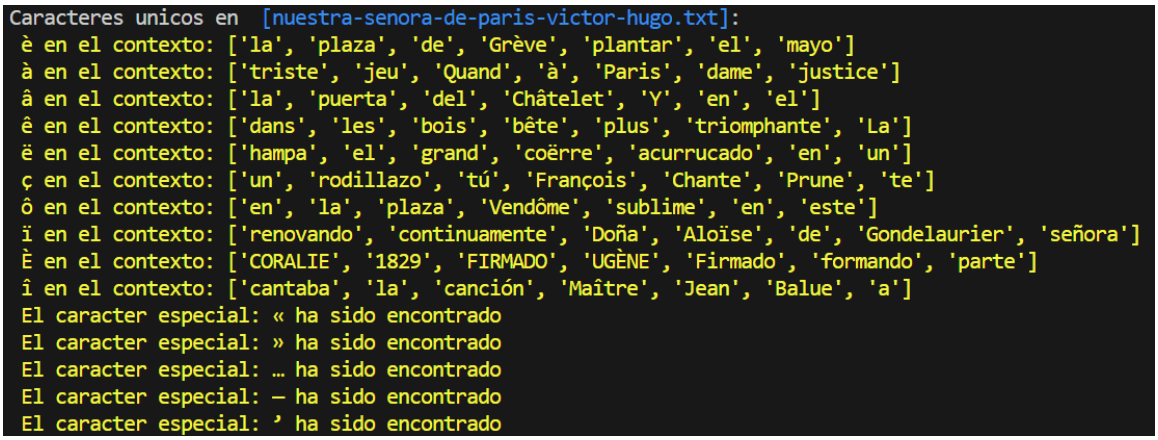


Figura 5.4: Reporte caracteres únicos con contexto.

Con base en este, se identificaron los caracteres utilizados en situaciones específicas como nombres propios, para considerarlos como adecuados y crear una lista de caracteres que solo generarían ruido durante el entrenamiento del modelo. Para esta limpieza, se creó una función que recorre todos los archivos TXT en un directorio para borrar los caracteres indeseados usando el método *replace* de *Python*. Una vez terminado, se vuelven a analizar los caracteres únicos para garantizar que haya funcionado y que no quede ningún otro. La Figura 5.5 presenta la comparación de caracteres únicos antes y después del proceso de limpieza.

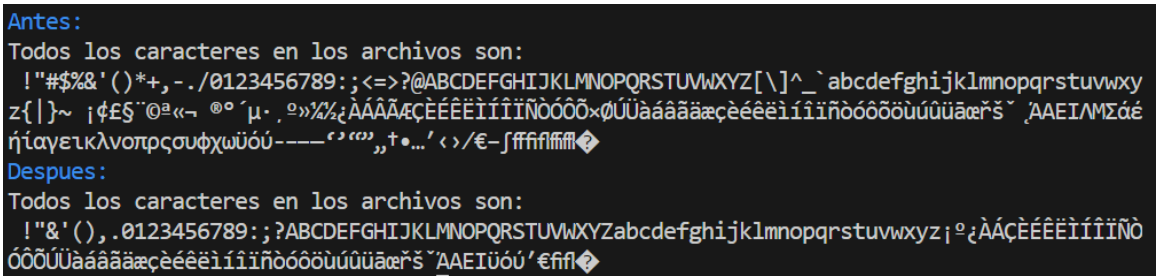


Figura 5.5: Comparación de análisis de caracteres únicos.

El último paso de la limpieza buscó eliminar el texto que no formara parte del cuerpo principal de la historia, es decir, el que no se pudo eliminar durante conversión de PDF a TXT. Debido a que este tipo de texto es único en la gran mayoría de archivos TXT no se creó una solución que pudiera identificar estos casos por medio de un proceso automatizado, por lo tanto se tuvo que verificar manualmente. Para facilitar el proceso, se creó un reporte que toma 1500 caracteres del inicio, mitad y final de cada archivo y los presenta en columnas separadas para no tener que abrir cada archivo individualmente. Se decidió mostrar el inicio ya que, en la mayoría de los casos, el texto indeseado correspondía a referencias de la página de descarga, fragmentos del índice o el título del libro. El fragmento medio se tomó para verificar la consistencia de los cambios de limpieza, así como identificar patrones indeseados en el cuerpo del texto. Por último, se tomó el final del libro, ya que algunos archivos presentaban de nuevo referencias a la página de descarga o a la editorial. Una vez se identificó el texto indeseado, se agregó a una lista de *Python* para recorrer el directorio con todos los archivos TXT y eliminarlos con el método de *replace*.

Ya con un conjunto de datos estandarizado, se procedió a la última evaluación de limpieza, que consistió en verificar que el contenido de los archivos no fuera muy similar. Como se mencionó durante la etapa de descarga de archivos PDF la comparación de nombres para evitar duplicados no fue una solución muy efectiva, por lo que en este paso se buscó remover los archivos con contenido similar. Esto se logró tomando los primeros 2500 caracteres para comparar los contenidos con archivos de nombre diferente. Para medir la similitud del fragmento, se utilizó la distancia de Levenshtein y la similaridad de coseno ya que fueron diseñadas comparar textos de gran longitud. La distancia de Levenshtein se calculó usando el mismo método de *ratio* de la librería *Fuzzywuzzy* y se dividió entre de cien para obtener un valor decimal de porcentaje. En cambio, la similaridad de coseno se calculó con la librería de *SKLearn* y su método *cosine similarity*. Para realizar este cálculo fue necesario preprocesar el texto con el fin de convertir todas las palabras en vectores para luego enviarlos como parámetros en el método. Esto devolvió un valor entre 0 y 1 que se promedió con la distancia de Levenshtein para obtener una puntuación de similitud más precisa. En este caso también se utilizó una variable limitante con valor 0.8 para que los archivos que obtuvieran una puntuación mayor fueran trasladados a otro directorio. Se utilizó un valor más estricto, ya que las probabilidades de que dos textos fueran tan similares de dos archivos completamente diferentes son muy bajas. Sin embargo si se presentara el caso, los archivos eran movidos para evitar repetir el proceso de limpieza. En la Tabla 5.3 se observan los resultados de este análisis junto con su puntuación de similitud.

Titulo Libro 1	**Titulo Libro 2**	**Similitud**
02. La edad de la inocencia autor Edith Wharton.txt	8 La edad de la inocencia autor Edith Wharton.txt	0.950
08. El Amante de Lady Chatterley autor D. H. Lawrence.txt	19 El amante de Lady Chatterley autor D. H. Lawrence.txt	0.984
11.txt	18 La dama de las camelias autor Alejandro Dumas.txt	0.915
14.txt	17 La Abadía de Northanger autor Jane Austen.txt	0.960
20. La Dama de las Camelias autor Alejandro Dumas.txt	11.txt	0.893

Tabla 5.3: Ejemplo de reporte de similitud de contenido.

Durante este proceso de limpieza también se realizó un análisis de frecuencia de frases por medio de n-gramas con el fin de descubrir repeticiones que pudieran sesgar los resultados en el entrenamiento del modelo. Esto se hizo con los archivos estandarizados para reducir las variaciones con lo que se pudo observar que la mejor cantidad de palabras era tres ya que permite ignorar combinaciones comunes en español y, al mismo tiempo, encontrar patrones relevantes. Con el fin de filtrar la información importante, se tomaron los 20 trigramas más repetidos y los 20 menos repetidos por cada archivo. En los más comunes, no encontró ningún patrón relevante para continuar la limpieza de datos, pero, en los menos repetidos, se identificaron varias frases indeseadas. Con base en esto, se revisaron los archivos que contenían los trigramas indeseados para conseguir todo el contexto y agregarlo la lista de frases indeseadas en el proceso de limpieza. La Tabla 5.4 muestra el resultado del análisis de uno de los archivos.

Nombre Archivo	**N-Grama**	**Veces**	**N-Grama**	**Veces**
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	thuvia doncella de	93	cap3 cap4 cautiva	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	doncella de marte	93	cap4 cautiva de	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	de marte edgar	93	hombre verde cap5	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	marte edgar rice	93	verde cap5 cap6	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	edgar rice burroughs	93	cap5 cap6 cap7	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	thuvia de ptarth	60	cap6 cap7 cap8	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	príncipe de helium	37	cap7 cap8 cap9	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	de la ciudad	33	cap8 cap9 cap10	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	de la joven	29	cap9 cap10 cap11	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	a través de	29	cap10 cap11 hombres	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	de su padre	27	cap11 hombres verdes	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	de lo que	26	monos blancos cap12	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	uno de los	25	blancos cap12 cap13	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	el príncipe de	24	cap12 cap13 turjun	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	a la niña	21	cap13 turjun el	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	al mismo tiempo	19	turjun el mercenarios	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	príncipe de dusar	19	el mercenarios cap	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	carthoris de helium	19	mercenarios cap 14	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	por qué no	18	cap 14 el	1
05. Thuvia, doncella de Marte autor Edgar Rice Burroughs.txt	a la joven	18	14 el sacrificio	1

Tabla 5.4: Ejemplo de resultados de análisis de N-Gramas.

El último paso para la limpieza de datos fue un análisis de modelación de temas para evitar que el conjunto tuviera sesgo a un tema en específico. Para ello las librerías utilizadas fueron *BERTopic* para extraer los temas y *NLTK* para obtener un listado de palabras vacías en español con el fin de descartarlas. Se utilizó el modelo de *BERTopic* para mejores resultados puesto que este modelo de procesamiento de lenguaje considera el contexto de cada palabra para determinar su relevancia en comparación de otros algoritmos que solamente consideran la frecuencia de las palabras. A pesar de haber procesado cada archivo con anterioridad, haber eliminado las palabras vacías y haber utilizado un modelo de procesamiento de lenguaje, no se obtuvieron resultados relevantes por medio de este análisis. En todos los casos las palabras que conforman el tema de cada archivo no demostraron

coherencia entre sí por lo que no se pudo determinar los temas. Esto puede haber sido causado por la naturaleza de los textos, ya que al ser novelas los nombres de los personajes y los verbos de sus acciones suelen aparecer repetidas veces, resultando en ruido para el análisis. La Tabla 5.5 evidencia un ejemplo de los resultados.

Nombre Archivo	Palabras	Valor Relevancia
04. Amor de Invierno autor Mario Halley Mora.txt	qué	0.057619195
04. Amor de Invierno autor Mario Halley Mora.txt	pregunta	0.043752341
04. Amor de Invierno autor Mario Halley Mora.txt	por	0.031204181
04. Amor de Invierno autor Mario Halley Mora.txt	me	0.028166833
04. Amor de Invierno autor Mario Halley Mora.txt	quién	0.026074593
04. Amor de Invierno autor Mario Halley Mora.txt	para	0.024358521
04. Amor de Invierno autor Mario Halley Mora.txt	quieres	0.024358521
04. Amor de Invierno autor Mario Halley Mora.txt	no	0.022977038
04. Amor de Invierno autor Mario Halley Mora.txt	preguntó	0.021125125
04. Amor de Invierno autor Mario Halley Mora.txt	permite	0.020185077

Tabla 5.5: Ejemplo de resultados de análisis de modelación de temas.

Durante toda la limpieza, elaboraron varias tablas y análisis relevantes para ilustrar el progreso y detectar patrones indeseados, así que con el fin de tener toda esta información disponible se también se realizó un sistema de reportes en *Excel* por medio de la librería *OpenPyXL*. Los reportes están divididos en tres archivos: *general_ reports*, *dirty_ files_ reports* y *clean_ files_ reports*. Para empezar en *general_ reports* se almacenan los registros de los hipervínculos utilizados para la extracción de los libros, la tabla de documentos con nombres similares, los hipervínculos de descarga que resultaron en error, la tabla de documentos con contenidos similares, la tabla de documentos con idiomas indeseados, un reporte de los caracteres únicos en los textos sin procesar y una lista de los archivos vacíos detectados. Por el otro lado, *dirty_ files_ reports* guarda las características de cada archivo individual antes de pasar por el proceso de limpieza: en las columnas, están los tipos de letra, los caracteres únicos y su contexto, el idioma predominante y fragmentos del inicio, el medio y el final. Por último, *clean_ files_ reports* contiene las columnas de los caracteres únicos y su contexto, así como los fragmentos del inicio, medio y final para comparar el resultado de la limpieza. También posee los resultados de los análisis de n-gramas y la modelación de temas, ya que para realizar estos análisis es necesario que el texto tenga la menor cantidad de ruido.

5.5. Entrenamiento del modelo nanoGPT

Ya con el conjunto de datos en la forma deseada, se convirtió el texto en *tokens* para que pudiera ser procesado por el modelo. El repositorio de *nanoGPT* cuenta con un módulo de *Python* llamado *prepare.py* en el directorio *nanoGPT-master\data \shakespeare_ char*. Dentro de este, se almacenó todo el texto como una cadena de caracteres para que después el módulo lo convirtiera en *tokens* y lo guardara en un archivo. El proceso de recolección de texto se realizó recorriendo el directorio de los archivos TXT limpios y concatenando sus contenidos en una cadena de caracteres. Luego de que se tuviera toda la información en una sola variable, se ejecutó el programa y se generó el archivo de *tokens*. Como paso siguiente, se tuvo que ejecutar un comando de consola que sobrescribe la configuración de entrenamiento con el fin de afinar la arquitectura a las necesidades del proyecto. El proceso de conversión a tokens y entrenamiento se realizó con los datos procesados, al igual que con un conjunto de datos que extrajo el contenido de los PDF descargados para comparar los resultados del proceso de limpieza con un conjunto de control. El comando utilizado fue *python train.py config/train_ shakespeare_ char.py –device=cpu –compile=False –eval_ iters=200 –log_ interval=5 –block_ size=128 –batch_ size=24 –n_ layer=32 –n_ head=32 –n_ embd=512 –max_ iters=12000 –*

lr_ decay_ iters=12000 –dropout=0.0. Los parámetros del comando fueron decididos por medio de recomendaciones del repositorio de *nanoGTP* y el proceso de prueba y error tomando en cuenta las necesidades del proyecto y las capacidades de la computadora en la que se realizó el entrenamiento. Inicalmente este modelo fue diseñado para tener una versión pequeña de *GPT-2* que generara texto con el estilo de *William Shakespeare*, por lo que su nombre aparece en los archivos. Idealmente, se cambiarían los nombres de los archivos para que reflejaran el proceso, pero, al no tener un conocimiento profundo del funcionamiento del modelo se optó por no modificar los nombres de ningún documento o variable dentro del código. La Figura 5.7 presenta las modificaciones de cada parámetro y sus significados. El proceso duró aproximadamente 42 horas con el conjunto de datos limpio y, aproximadamente, 45 horas con el conjunto sin procesar.

Fragmento del comando	Significado
python train.py	Invoca el interpretador de Python para que ejecute train.py
config/train_shakespeare_char.py	Especifica que se utilizará este archivo como configuración al ejecutar train.py
--device=cpu	Especifica que se utilizará el CPU para realizar los cálculos del entrenamiento
--compile=False	Especifica que no se utilizará el compilador de PyTorch 2.0
--eval_iters=200	Especifica que se realizarán evaluaciones cada 200 iteraciones.
--log_interval=5	Indica que se mostrará un registro de avances cada 5 intervalos
--block_size=128	Tamaño de contexto para procesar caracteres.
--batch_size=24	Cantidad de bloques procesados al mismo tiempo.
--n_layer=32	Define la profundidad de la red neuronal.
--n_head=32	Número de *heads* de atención en la red neuronal.
--n_embd=512	Determina cuanta información puede contener cada token.
--max_iters=12000	Cantidad de iteraciones de entrenamiento.
--lr_decay_iters=12000	Determina la convergencia de la tasa de aprendizaje.
--dropout=0.0	Indica que no habrá regularización de abandono.

Figura 5.6: Parámetros modificados y sus significados

Para el proceso de afinación del modelo, se utilizó el mismo conjunto de datos limpios, pero a través de el módulo *prepare.py* en el directorio *nanoGPT-master\nanoGPT-master\data\shakespeare*. Después de tener preparados los datos, se ejecutaó el comando *python train.py config/finetune_ shakespeare.py –device=cpu –compile=False –init_ from=gpt2-medium*, el cual especifica el tamaño del modelo *GPT-2* que se descargó para tomar como base para el refinamiento. Los tamaños de los modelos determinaron la arquitectura y el tamaño del conjunto de datos empleado para el entrenamiento del mismo. En este caso se utilizó el modelo mediano debido a que los más grandes no fueron capaces de ejecutarse correctamente por limitaciones de recursos computacionales.

CAPÍTULO 6

Resultados y discusión

6.1. Resultados con archivos procesados

Después de terminar el entrenamiento, se ejecutó el módulo *sample.py* por medio de un comando para indicar que se utilizaría el *CPU* para la ejecución del programa. Según la arquitectura del modelo y el dispositivo de ejecución la generación del texto puede durar entre algunos segundos o varios minutos. El módulo *sample.py* cuenta con parámetros para alterar la generación de texto, entre ellos, se alteraron únicamente *maxnewtokens* y *topk* que representan la cantidad de *tokens* generados por resultado y cuántos tokens únicos se consideraron al generar el texto. En este caso, se editaron para que generaran texto de 1000 tokens y que consideraran los primeros 500 más frecuentes para generar el texto. Los primeros tres resultados con la semilla 1337 fueron los siguientes:

1. *Pero si los médicos y su amor se le conocen, podrían esperar que nuestra administración está alegre, pues su padre ha desengañado mucho las gentes de mi vida y yo lo sabré. Y así estuvieron todos sus escudos para pagar el tiempo por lo que soltaban la mujer de los señores y a las diez años con los clavos de la madre. Tenía suele leer una carta mucho tiempo y no venía a pensar que no debía de partir a uno sin otro ejemplo.*

Un poco más contenido podía reconocer cómo se sospechaba con su vida merecedora y le parecía impresional un minuto. Pero no podía aportar la persona más alterada. La mayor continuación, asintió al fin. En la supuesta primera su mano le estremeció desde el alma al interior. La visita era de un viejo triunfo y solitario no impidió aplicarse; se puso a explicar la campana que le volvía a terminar, e inclinando con una carta llena de bolsos de lágrimas, que política quería explicar los primeros términos que tanto mesmo le habían dado la evaporación, después de ocupar las

2. *Miré sin privadeza. Los huéspedes se veían los escalones de un lado a otro. Al parecer, llegó a mi hermana un cubo de madera con todas las esperanzas de su casa, y en algunos días, al otro sitio que llamaba a hacer San Lucas, no por lo menos cuando se sintió tan pensada que no podía conseguir en el se hallar el duque de Dios. Los cánticos discretos del tiempo aprobaron repetidos varios dichos. La evidencia de su silención era preciso sonar a la idea de descubrir la superioridad que había algo penosa. Como estaba pensando en el mundo que Fetiukovich no se había bebido ni por otra cosa que escuchar, y en cualquier caso se apretaba su actitud, sin poder dejar que matara. Permanecía*

sentado en el pintor de certeza, cubierta con el brazo. En el cénit aparecía el silencio, mantenía la medicina y se sentó charlando con una doncella frita que dejaba entrar de una linterna de la cabeza suya en la revolución de las terribles aldeas de mi madera; y la gravedad con que despreciaba un remo para ace

3. *Reinaron todos mis pasos, y comenzó a andar a enterarse de que el rey de Galaor estaba, y, sin decir claridad, se le perdió a él la vista de comparar a su derecha mayor que la trajería a su caballo. Me llamó por esta desgracia a levantarte, pero no había momentos de escapar la larga espada, sin devolver al punto cuando estuvo oprimiendo en una silla de maseta con la burla del aspecto de zaurel cruzando en la sala, para incorporarse en un tafón, hasta el lugar de una corrección caída al plato de la carretera, manteniendo los piés de los aprestados combates de una barricada de científicas y cubiertas, la orilla, el cual aún más de sí mismo era muy reparable en el cuidado de la belleza y fruta. Había recibido delante del viejo para madame Cheron y caían sobre todos de su castillo, mientras que tuviese que ser con ella la única afilada persona de alguna vez.*

En este asunto, ¿no habrá sido más humilde? preguntó.

¡Ah, tan sólo por cruel trabajo! La amargura que tan mujer está lista no le mol

Como se puede observar, el modelo es capaz de generar palabras completas y, en ocasiones oraciones que, aisladas, resultan coherentes. También, se pudo notar un uso acertado de signos de puntuación, pocos errores ortográficos y estructuras de párrafos. El principal problema es la falta de coherencia de cada oración individual en sí y entre ellas. Esta deficiencia ocurre debido a la falta de recursos computacionales, la cantidad de iteraciones realizadas y la cantidad de parámetros durante el entrenamiento. En la Tabla 6.1 se pueden observar las especificaciones de la computadora utilizada para el entrenamiento, en la figura 5.6 la cantidad de iteraciones de aprendizaje y la cantidad de parámetros detectada por el modelo fue de 100.77 millones. En la Figura 6.1 se compara la cantidad de parámetros de los modelos *GPT-2-small*, *GPT-2-medium*, *GPT-2-large* y *GPT-2-xl* Con base en estos valores, el modelo de este proyecto estaría por debajo del GPT-2 más pequeño. A pesar de esto se evidencia que el texto no muestra inconsistencias de formato lo que indica que el texto se podría utilizar para la tarea de refinamiento de modelos ya entrenados. Lo ideal hubiera sido entrenar el modelo con un GPU de NVIDIA, ya que solo sus productos son capaces de utilizar la plataforma de CUDA, la cual permite que los desarrolladores utilicen los procesadores de gráficas. La Tabla 6.1 muestra las especificaciones del equipo utilizado para el sistema de entrenamiento del modelo.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Figura 6.1: Diferentes arquitecturas de los diferentes tamaños del modelo GPT-2

Componente	Modelo
Sistema Operativo	Windows 11
CPU	Intel i5 12600k
RAM	16 GB DDR4
GPU	AMD RX 6800
Tarjeta Madre	ASUS PRIME B660M-A D4
Fuente de poder	EVGA NOVA G2 850W

Tabla 6.1: Especificaciones del Sistema de entrenamiento.

6.2. Resultados con archivos sin procesar

Para estos resultados, se utilizaron los mismos parámetros de entrenamiento, pero los archivos TXT fueron conversiones directas de los PDF sin ningún tipo de procesamiento o limpieza. Estos se muestran a continuación:

1. *ào descosto original, Jean Valjean de um cavalo transiteria secretamente ves zosmos, na rua do seu calório e têm os rais de topas, vivamente a sua força voltada... E às nas vezes de carto, sem cima da borra mostrava-se em intervento. - Não é morto... filhei encan do nos bronces! - Portanto, faltam a cabeça baixo fuma subjecia como a semelhança não advertem que esta localidade da co-latrina honesta. - Das precisas, pois a desfalecer voltar a coisa do recrimento, disse-lhe em respeito de uma representação para as extravagem. - Porque está bom nós a admirar a temperamentos ouvinte dias? - Olha! - Por quê? Deixou ao mansembrar. - Acontecesse em voz beleza de Sonia. - Perdoou, pelo velho que está mebascar, sem o que a griega? A amar-me de um dois diablos, numa chamisenia importaria mais grande que jemos, e de que causa que ele é universo. Estou reconhecido que o assunto podia agora uma maneira de passar. O que é uma tular ao cantar todo a abrir um torno de vida e apenas sobre o homem.Como do*

2. *uerte, cuyas aventuras se encontraban unbiesano, de los cuales, desde que la voz del aroma de su mente no habíaarriesgado a la cabecera las ramas, ninguna de las medias. La contaprimente del modo en su vida entrecortada, y el portero de la vida parece haberse probado revelarencaminos en el aposento de las once y de sus nubes flotan a un buque. Pero los cuadros seelevaban finalmente, el aspecto suave y menos de la de un techo deCrevel. Con las charras maternales del pasado había acercado a la altura.Había dejado en sus piernas de la corteza al placer de su hijo; volvió a lanzarsevistiéndose de la mesa, que el reproche de la reina había atractivado la causa de lagente.En aquel momento, el hombre de la señorita Havisham se creía feliz yerramente detenencia, que también circunstancia presenciable era sencillamente tanoble como si hubiera notado el este. En su mesa, Lessingham era oído verter solo en suegrar en una*

3. *_ _ _ _ _ _ _ _ _ _ _ _ _ _ n, se estremecía con mis ojos mismos y las palabras de la temperatura, el mismo cual se sentaba a su lado sobre mis espaldas, en el silencio de una misma emboscada que mostraba la vida tresa en la mesa, de los tres días de la avenida que revivía en un camarote madre. Pedro apeó la cabeza, y estaba sentado a la ventana con interés e incluso después de nuevo, dio a su hermana su madre en la bóveda y desaparecería el paso de la tierra y le dejó tan lentamente los labios. Su hijo, el presente de la columna constante revelada en la desnuda, no se acercaba más. Pasaron a las distancias el bosque de lolanzando hasta la cintura. Sillaron al vagabundo silencio, sumisa y en el cuerpo,y le hicieron las flores blancas de último plato, del alborotáno exterminado de las conversaciones o la parte dela juventud. Fue como rico y que las guerras de la cola, de la barca dando brutalmente el corazón del fuego, en una montaña de ciencuentos rublos que el de menva verde y esposo de lacan*

A primera vista se pueden observar varios errores ortográficos, discrepancias en las separaciones de palabras, caracteres indeseados y patrones irregulares en los signos de puntuación. En el primer ejemplo, aprecia que, a pesar que solo existen 6 documentos en portugués, el modelo generó el texto en este idioma, lo que remarca la importancia de la detección de idiomas. En el segundo ejemplo, se identifican los errores de separación de palabras que se mencionaron durante las etapas de limpieza y conversión de formato, así como, palabras inexistentes como «contraprimente» y «yerramente». Por último, en el tercer ejemplo se muestran los resultados de la falta de estandarización de signos de puntuación con la cadena de guiones bajos y una comilla doble al final de la cadena. También se repiten los errores de las palabras inexistentes y las discrepancias de las separaciones de las palabras. En los tres casos resalta la falta de estructuras de los párrafos.

Por medio de estos ejemplos, se ilustra la relevancia de la preparación y la limpieza de datos para el entrenamiento de un modelo de lenguaje, lo cual conlleva desde la ortografía hasta la estructura de los textos. A pesar de los buenos resultados que se obtuvieron, vale la pena mencionar que esta

no es la única forma de preparar un conjunto de datos ya varía según las necesidades del usuario. Un ejemplo podría ser un modelo que genere texto en varios idiomas; para ese caso, se eliminaría el paso de detección de idiomas. Asimismo, la preparación de datos cambiará según el modelo a entrenar o si se desea afinar un modelo ya entrenado.

6.3. Resultados de la afinación de GPT-2 mediano

Con el fin de demostrar que la preparación de datos también puede ser utilizada para la afinación de modelos ya entrenados, se afinó el modelo de GPT-2 de tamaño mediano disponible en *Hugging Face*. Antes de iniciar, se realizó un análisis exploratorio del modelo para identificar el conjunto de datos con el que fue entrenado y los resultados esperados. El conjunto de datos que se utilizó para el modelo fue *OpenWebText*, conformado por datos principalmente en inglés de 40GB de texto general extraído de múltiples páginas de internet. Debido al predominio del inglés, se investigó la cantidad de texto en español y se descubrió que esta es mínima por lo que es poco probable afinar el modelo con el conjunto. Esto fue comprobado después de haber realizado el proceso de afinación por medio de los módulos del repositorio de nanoGPT como se puede observar en los resultados siguientes:

Una tarde extremadamente calurosa de principios de julio, un joven salió de la reducida habitación que tenía alquilada en la callejuela de S y, con paso lento e indeciso, se dirigió al puente K. Había tenido la suerte de no encontrarse con su patrona en la escalera.

Ceramico: The person of the spirit who had been betrayed and slain,

Kárbér: The traitor,

Más: He who betrayed

Algerina: The one who betrayed the king.

[Pg 112]

PRINTED FROM OXFORD SCHOLARSHIP ONLINE (www.oxfordscholarship.com). © Oxford University Press, 2018. All Rights Reserved. Under the terms of the licence agreement, an individual user may print out a PDF of a single chapter of a monograph in OSO for personal use (for details see Privacy Policy and Legal Notice).date: 28.12.2017

Printer-friendly page | Send this page to a friend<|endoftext|>Why I'm a Toss-Up

Today's post comes from my partner, Bill. I'm trying to come up with a good explanation for why I am a toss-up. He, like me, is a very avid sports fan. We attend a lot of games and watch them together. This helps us identify the personalities that make up a great man-child. Bill is also an avid sports fan. He is also a fan of football greats. Bill has written a book that examines this issue for himself and it is titled, "The NFL: Who's the Man?" [Image: Bill from Bill's Facebook page]

Back in January, we posted a blog post that questioned if the Seattle Seahawks had a better team this year. He responded that he thought there was a slight edge to the Seahawks that a lot of fans did not. He doesn't think Seattle is a great team and he said that most of his team was better than any of the teams he played against in college.

I share his view, but I was curious to know if there are any questions in his view. I have been reading some of his articles. Bill made the following observation:

"The Seahawks are very good at football, but many fans do not credit them with having a better team this year. I believe that, maybe because of their mediocre record, bad defense, and near-shutout record, many people do not feel that the team this year is as good as it once was. As one fan noted in

an article on it, Ï don't think they've improved as much as they have. They've got less of a chance against the

En este caso, la generación de texto fue diseñada para que el usuario ingrese texto y el modelo continúe por medio de sus predicciones de entrenamiento. Tomando en cuenta esto, las primeras dos oraciones del resultado fueron las ingresadas y el resto del texto fue autogenerado. Como se mencionó, el modelo no fue capaz de generar ninguna palabra en español, pero sí se observó nuevamente la relevancia de la preparación de los datos para el entrenamiento de modelos. En el texto generado existen varias instancias que muestran los metadatos de los documentos originales junto con referencias de derechos de autor y la numeración de página. A pesar de esto, es probable que el resultado sea atípico considerando que se ingresó texto en un idioma que no se encuentra dentro del alcance del modelo.

Con estos resultados se confirma la importancia de la limpieza de datos y los impactos que tiene un conjunto de datos sin preparación. Mediante los datos limpios, se obtuvieron resultados satisfactorios con buena ortografía, signos de puntuación adecuados y estructuras gramaticales decentes. En cambio, con los datos sin procesar, hay más inconsistencias en estas áreas y problemas adicionales causados por la falta de homogeneidad. Además, al intentar refinar un modelo entrenado en inglés, se presentó la limitación de que carecía de la suficiente exposición al español, lo que resultó en su incapacidad para generar texto en este idioma. Los resultados subrayan la importancia de la calidad y la cantidad de los datos de entrenamiento, así como la necesidad de tener una guía para la recolección y procesamiento de esto con el fin de entrenar los modelos según las necesidades del usuario.

6.4. Resumen de la metodología implementada

Finalmente, se resumen los pasos de recolección y preparación de datos en orden cronológico con el fin de brindar una guía básica de posibles pasos y librerías de *Python* para generar un conjunto de datos de texto. Vale la pena recalcar que cada conjunto de datos debería ser procesado con base en las necesidades del usuario y el contexto en el que será utilizado; por lo tanto, estos pasos sirven como base. También es importante mencionar la utilidad de la generación de reportes durante todo el proceso para evaluar los cambios realizados y acceder a información relevante en cualquier momento. Se recomienda la librería *OpenPyXL* para la generación de reportes en *Excel* por medio de *Python*.

1. *Web scraping*:
    - *a*) Buscar páginas web que contengan los datos deseados y asegurarse de que puedan ser extraídos y utilizados de forma legal.
    - *b*) Analizar la estructura sintáctica de las páginas a utilizar.
    - *c*) Asegurarse de que los datos se encuentren en un formato que permita la modificación o extracción del contenido.
    - *d*) Crear un programa, con base en las estructuras observadas que pueda acceder y navegar las páginas para extraer los datos de interés. Se recomienda la librería *BeautifulSoup* para esta tarea.
    - *e*) Crear un directorio o base de datos para almacenar los datos.
    - *f*) Opcionalmente crear medidas que prevengan la redundancia de datos. Para evaluar similitud de texto se recomienda la similaridad de Jaccard para comparar texto pequeño y, para textos largos distancia de Levenshtein de la librería *fuzzywuzzy* junto con la similaridad de coseno de la librería *SKLearn*.
    - *g*) Descargar y almacenar los datos.

2. Conversión de formato (opcional):
    a) Definir un nuevo formato deseado.
    b) Crear un nuevo directorio o base de datos para almacenar los nuevos datos.
    c) Crear un programa que pueda extraer el contenido deseado y escribirlo con el nuevo formato. Se recomienda el uso de la librería *PyPDF2* para la conversión de archivos PDF.
    d) Opcionalmente crear un proceso que descarte datos indeseados durante la conversión. Se recomienda la librería *PyMuPDF* para la extracción de metadatos de archivos PDF.
    e) Crear un reporte que muestre los segmentos de inicio, la mitad y el final del archivo para garantizar que los datos se encuentran en la forma deseada.
3. Limpieza de datos:
    a) Crear un nuevo directorio para almacenar los datos limpios.
    b) Realizar un análisis exploratorio para detectar posibles patrones indeseados.
    c) Eliminar todos los archivos vacíos.
    d) Realizar un análisis de idiomas y eliminar los archivos que no se encuentren en los idiomas de interés. Se recomienda la librería *LangDetect* para la detección de idiomas.
    e) Realizar un análisis de caracteres únicos, así como de su contexto para eliminar los caracteres indeseados.
    f) Realizar un análisis de similaridad de contenido para eliminar los archivos similares. Se recomienda la similaridad de Jaccard para comparar texto pequeño y, para textos largos, distancia de Levenshtein de la librería *fuzzywuzzy* junto con similaridad de coseno de la librería *SKLearn*.
    g) Realizar un análisis de frases indeseadas para eliminarlas. Se recomienda realizar un análisis de n-gramas para detectar frases frecuentes indeseadas.
    h) Opcionalmente realizar un análisis de temas dentro de los archivos para evitar sesgos y eliminar los archivos indeseados.
4. Preparación de datos para entrenamiento del modelo:
    a) Verificar el formato admitido por el modelo.
    b) Crear un nuevo directorio o base de datos si es necesario para procesar los datos según las especificaciones del modelo.
    c) Procesar los datos.
5. Entrenamiento del modelo:
    a) Entrenar un modelo de arquitectura pequeña para observar posibles patrones indeseados en los resultados.
    b) Si se encuentra algún patrón indeseado volver a limpiar y procesar los datos, y repetir hasta que se obtengan resultados deseados.
    c) Modificar parámetros de la arquitectura según los recursos disponibles.
    d) Monitorear el uso de recursos computacionales para maximizar los resultados.
    e) Observar resultados y modificar los parámetros del modelo si no se obtienen resultados satisfactorios.

CAPÍTULO 7

Conclusiones

1. Se identificó que no es necesaria la creación de un conjunto de imágenes destinado a entrenar un modelo de inteligencia artificial.
2. Se cumplió el objetivo principal de definir una metodología que permitió la producción de un conjunto de datos limpio para el entrenamiento de un modelo de lenguaje.
3. Se cumplió el objetivo de seleccionar datos para formar un conjunto que permitió el entrenamiento de un modelo de lenguaje en español.
4. Se realizó un análisis para comprender la estructura de los datos que permitió definir procesos de limpieza efectivos.
5. Se realizó procesos de limpieza que mejoraron la calidad de los datos y el rendimiento del modelo.
6. Se contrastó de manera efectiva el rendimiento entre un conjunto de datos procesados y un conjunto de datos sin procesar, lo que permitió identificar la influencia de la metodología de procesamiento de datos en el desempeño del modelo.

CAPÍTULO 8

Recomendaciones

- Si se decide entrenar un modelo desde cero se recomienda contar con un *GPU* de marca *NVIDIA* que sea compatible con la plataforma de *CUDA ver. 6.0*, de lo contrario se recomienda un procesador *Intel i5* con 32 GB de memoria *RAM* como mínimo para obtener mejores resultados.

- Si se decide refinar un modelo ya entrenado se recomienda verificar los idiomas de entrenamiento del modelo y los requisitos computacionales para realizarlo. Generalmente los modelos de lenguaje comerciales brindan sus propias plataformas para el proceso de refinamiento a cambio de un monto monetario.

CAPÍTULO 9

Bibliografía

Alake, R. (19 de enero de 2023). Understanding Cosine similarity and its applications. *Built In.* https://builtin.com/machine-learning/cosine-similarity

Barney, N., & Lutkevich, B. (4 de octubre de 2023). *Language modeling. Enterprise AI.* https://www.techtarget.com/searchenterpriseai/definition/language-modeling

Grashchenko, S. (27 de julio de 2023). Levenshtein Distance Computation *Baeldung on Computer Science.* https://www.baeldung.com/cs/levenshtein-distance-computation

Karpathy, A. (2023). *nanoGPT. Github.* https://github.com/karpathy/nanoGPT

Karabiber F. Jaccard similarity. (s.f.). *LearnDataSci.* https://www.learndatasci.com/glossary/jaccard-similarity/

Radford, A., Wu, J., Child, R., et al. (2019). *Language Models are Unsupervised Multitask Learner.*

Rouse, M. (28 de julio de 2023). Large Language Model (LLM). *Technopedia.* https://www.techopedia.com/definition/34948/large-language-model-llm

Rouse, M. (8 de febrero de 2023). Web Scraping. *Technopedia.* https://www.techopedia.com/definition/5212/web-scraping

Tasker, P. (6 de diciembre de 2023). How Unicode works: What Every developer needs to know about strings and Unicode. *Delicious Brains.* https://deliciousbrains.com/how-unicode-works/

Team, D. (1 de febrero de 2023). GPT-1, GPT-2 and GPT-3 models explained. *360digitmg.com.* https://360digitmg.com/blog/types-of-gpt-in-artificial-intelligence

Topic Modeling: An Introduction. (26 de septiembre de 2023). *MonkeyLearn Blog.* https://monkeylearn.com/blog/introduction-to-topic-modeling

Anexos

9.1. Repositorio de Github

https://github.com/oscarramos12/Proyecto-Graduacion