



Intelligent Data Analysis Laboratory

TRABAJO FINAL DE MASTER INTELIGENCIA ARTIFICIAL AVANZADA Y APLICADA



CRITERIOS ESG

MEDIANTE TÉCNICAS AVANZADAS DE NLP
PARA APLICACIONES EN ECONOMÍA

AUTORES

ÓSCAR MATEOS LÓPEZ

RAFA DOMÉNECH SERRANO

MÁXIMO MEGÍAS MATEOS

VALENCIA

15 SEPTIEMBRE 2022

BREVE DESCRIPCIÓN

El presente trabajo describe como se ha llevado a cabo un proyecto NLP desde la captura de datos hasta el despliegue a producción. Se presenta el estudio, las pruebas, el desarrollo de una API capaz de calcular los criterios ESG de compañías del SP500 a partir de artículos y noticias. Para ello se han utilizado modelos Transformers (variantes de BERT, Pegasus, T5) para las diferentes piezas del pipeline que conforma el proyecto desarrollado. Estos modelos se sitúan en el estado del arte. Para cerrar el trabajo se ha desplegado a producción una webApp que consulta la API desarrollada y muestra de forma gráfica los resultados.



VNIVERSITAT
DE VALÈNCIA

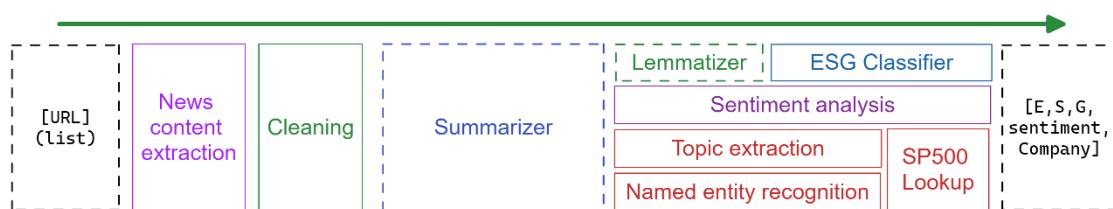


0. RESUMEN

La elaboración de informes ESG se viene haciendo de forma mayoritariamente humana y manual, según la información disponible públicamente. Aunque también es cierto que en los últimos años han proliferado proveedores que ofrecen criterios ESG mediante IA, es cierto que no hay transparencia en la información de los modelos utilizados. El presente trabajo se plantea para solventar los problemas de sesgo y coste que implica cualquier proceso humano especialmente en esta métrica que, conforme se va implantando su uso, tiene repercusiones directas, como por ejemplo en la capacidad de las empresas para recibir inversión.

El objetivo principal del presente trabajo es la automatización de la generación mediante técnicas avanzadas de NLP de los criterios ESG a partir de noticias y su asociación de estos valores a las empresas que aparecen en dichas noticias. Para llevar a cabo las tareas de generación de resúmenes, reconocimiento de entidades, análisis de sentimiento y clasificación se han realizado finetunings de modelos Transformers que se sitúan en la parte alta del estado del arte.

El siguiente diagrama representa todas las piezas del modelo en las que se ha trabajado para conseguir dicho objetivo:



Entre los modelos NLP que han sido probados para resolver las diferentes piezas se encuentran desde el Textrank hasta el BERT, T5 y Pegasus pasando por modelos secuenciales LSTM. De todos estos modelos se pueden consultar resultados de los entrenamientos realizados como sus métricas. Además de las métricas típicas de *accuracy*, *precision* y *loss* también se ha utilizado la métrica Rouge para medir la calidad de los resúmenes. Como resultado se ha conseguido realizar un finetuning del modelo Pegasus de Google y conseguir una Rouge1= 44.2881 situándose en el top 1 en la web paperswithcode.com para la tarea de generación seq2seq.

En la tarea de clasificación de noticias y como pieza definitiva se ha obtenido una *accuracy*= 0.93 y precisión=0.83. Se ha realizado el despliegue a producción del modelo mediante una API de Hugging Face y el desarrollo de una webapp usando streamlit que explota dicha API.

Toda la información relativa a este proyecto en:

<https://huggingface.co/ESG-TFM-UV>

<https://github.com/rdose/tfm-main>





1. ÍNDICE

0. RESUMEN	2
1. ÍNDICE.....	4
2. PRESENTACIÓN DEL EQUIPO.....	7
3. MOTIVACIÓN DEL PROYECTO	8
ANTECEDENTES	8
MOTIVACIONES	8
JUSTIFICACIÓN.....	9
4. INTRODUCCIÓN.....	10
¿QUÉ ES ESG?	10
¿PARA QUÉ SIRVEN LOS CRITERIOS ESG?.....	10
CRITERIOS AMBIENTALES ESG	11
CRITERIOS SOCIALES ESG.....	11
CRITERIOS DE BUEN GOBIERNO ESG	11
LA INVERSIÓN SOSTENIBLE Y SU CLAVE EN ESG.....	11
¿QUÉ NO ES ESG?	12
5. PREGUNTA DE NEGOCIO PLANTEADA	13
¿CUÁLES SON LOS DESAFIOS DE LOS ANÁLISIS ESG A RESOLVER?	13
6. ESTADO DEL ARTE DEL PROCESADO DEL LENGUAJE NATURAL PLN.....	15
ESTADO DEL ARTE. TÉCNICAS NLP.....	15
7. DESCRIPCIÓN DEL DATASET DE PARTIDA.....	19
8. DESCRIPCION DEL FLUJO DE TRABAJO.....	21
9. EXTRACCIÓN DE DATOS	22
WEB SCRAPING	22
EXTRACNET	22
10. ANÁLISIS Y PREPROCESADO DE LOS DATOS.....	24
PRIMER ANÁLISIS: ESTRUCTURA, CONTENIDOS Y DUPLICADOS.....	24
SEGUNDO ANÁLISIS: BREVE ANÁLISIS DE DISTRIBUCIÓN DE LAS CLASES ESG.....	25
TERCER ANÁLISIS: ANÁLISIS DE LONGITUDES EN CANTIDAD DE PALABRAS.....	25
LIMPIEZA DE DATOS	28
11. GENERACIÓN DE RESÚMENES	31
PUROS EXTRACTIVOS.....	32
PUROS ABSTRACTIVOS	32



TRANSFORMERS	36
MODELOS PRE-ENTRENADOS (SIN FINE-TUNING)	36
MODELOS PRE-ENTRENADOS (FINE-TUNING).....	37
12. ANÁLISIS DE SENTIMIENTO	45
13. EXTRACCIÓN DE EMPRESAS	46
MODELO PARA EXTRAER LOS NOMBRES DE LAS EMPRESAS.....	46
OBTENCIÓN DE DATASETS DE ENTRENAMIENTO	47
ETIQUETADO CON DOCCANO	47
ENTRENAMIENTO DEL MODELO SPAN CATEGORIZER.....	47
BUSQUEDA FLEXIBLE MEDIANTE FUZZY SEARCH	53
TOPIC MODELING PARA ASOCIACIÓN SEMÁNTICA.....	54
MODELO DE TOPIC MODELING CON BERTOPIC.....	55
OBTENCIÓN DE DATASETS DE ENTRENAMIENTO	55
ENTRENAMIENTO DEL MODELOS TOPIC-MODELING	55
14. CLASIFICADOR ESG	60
IMPLEMENTACIÓN	60
ENTRENAMIENTO E HIPERPARAMETROS.....	61
MONITORIZACIÓN.....	62
TASA DE APRENDIZAJE	62
RED NEURONAL DENSA	63
MODELO TRANSFORMER	64
PREPROCESADO Y LIMITACIÓN DE LOS DATOS DE ENTRADA.....	65
TRUNCADO	66
MÉTODO JERÁRQUICO	66
REDUCCIÓN UTILIZANDO TRANSFORMER	66
RESULTADOS.....	67
PRUEBAS PRELIMINARES DEL CLASIFICADOR ESG	68
15. DESPLIEGUE A PRODUCCIÓN	70
BACKEND API	70
EXTRACCIÓN DE CONTENIDOS	71
CLASIFICADOR ESG	71
ANÁLISIS DE SENTIMIENTO	71
EXTRACCIÓN DE EMPRESAS	71



FRONTEND	72
PUESTA EN MARCHA	72
FRONTEND	72
BACKEND	74
16. CONCLUSIONES Y POSIBLES MEJORAS	76
17. REFERENCIAS	77
33. LISTADO DE ILUSTRACIONES Y TABLAS	79
ILUSTRACIONES	79
TABLAS	80



2. PRESENTACIÓN DEL EQUIPO

El presente trabajo de final de máster ha sido desarrollado por tres alumnos de la primera promoción del master en Inteligencia Artificial Avanzada y Aplicada perteneciente a la Universitat de València impartido a partir de septiembre de 2020. El equipo se conformó por el interés común en técnicas de aprendizaje por refuerzo (RL), aunque al final no pudo ser y el presente trabajo versa sobre técnicas de procesado de lenguaje natural (NLP). Los tres integrantes del equipo son:

- Óscar Mateos López, titulado en Ingeniería Superior en Informática por la Universidad Autónoma de Madrid y Máster en Minería de Datos por la Universidad Complutense de Madrid (oscar.mateos.lopez@gmail.com)
- Rafa Domenech Serrano, titulado en Ingeniería Superior en Electrónica por la Universitat de València (radosse@alumni.uv.es)
- Maxi Megías Mateos, titulado en Ingeniería Industrial Superior en Automática y Electrónica por la Universitat Politècnica de València. (maxi.megias@gmail.com)

Debido a la problemática de la distancia y la incompatibilidad de horarios entre los integrantes. El trabajo se ha llevado a cabo realizando reuniones telemáticas periódicas durante 4 meses a horas intempestivas y utilizando plataformas online colaborativas para facilitar en la medida de lo posible la ejecución del mismo. Así que, dado que el máster también se impartió en modalidad online los integrantes aún no se conocen en persona. Es algo que ha quedado pendiente y que habrá que solucionar en breve.



3. MOTIVACIÓN DEL PROYECTO

En este punto del proyecto se definen los antecedentes del proyecto, la motivación con la que se afronta, así como su justificación y el porqué de la realización del proyecto.

ANTECEDENTES

El presente proyecto supone el primer proyecto basado en técnicas de procesado de lenguaje natural, de ahora en adelante usaremos su acrónimo en inglés NLP, sin tener en cuenta los realizados durante el curso al que se afronta el equipo de desarrollo.

Ninguno de los componentes posee conocimientos previos al proyecto acerca del concepto ESG “Medio Ambiente, Social y Gobernanza”, de su acrónimo en inglés “Environment, Social & Governance”. Tanto es así que la primera vez que el equipo escucha el concepto es en la primera reunión con Alberto Oteo, profesor de RL del máster.

El criterio o criterios ESG no son nuevos, hay artículos sobre la temática que datan del año 2008. Es cierto que se está poniendo de moda en el sector financiero, tanto es así, que tarde o temprano se deberán llenar formularios dejando ver el criterio ESG de cada cliente para realizar operaciones financieras con los bancos. Pero también es cierto que existe un proteccionismo con la información en el sector financiero, de manera que no se han encontrado publicaciones de los modelos de inteligencia artificial, de ahora en adelante IA, en todo caso si existen. Sólo se han encontrado artículos y noticias de prensa haciendo eco de la relevancia de los ESG de cara a las inversiones financieras a las que se hará referencia durante la presente memoria.

MOTIVACIONES

Supone una motivación personal y académica en especial saber que para finalizar el máster y conseguir el título propio del Master de Inteligencia Artificial Avanzado y Aplicada, de acuerdo con la normativa vigente, se necesita realizar con éxito el Trabajo Final de Máster. En el cual, además de aplicar los conocimientos adquiridos a lo largo del máster se realizarán estudios e investigaciones para profundizar en los conocimientos de las técnicas más avanzadas de NLP y su aplicación en concreto al problema que añade al trabajo y en general a la gran aplicabilidad que tiene NLP. Dada la transversalidad del NLP se puede decir que su campo de aplicación es universal, habiendo aplicaciones en la mayoría de sectores. Y es realmente en la realización de este trabajo donde el equipo se va a enfrentar a un problema real con todas las implicaciones que esto conlleva, la motivación también viene por esta parte dado que es dónde se aplican conocimientos y metodologías hasta ahora no usados, más cercanos al mundo laboral.

Por otra parte, también existe la motivación profesional que supone saber que el mercado actual está demandando profesionales con las competencias adquiridas durante el máster y que han quedado patentes durante el desarrollo y el resultado del presente trabajo. Desde el



inicio del trabajo, ha sido algo común a todos los integrantes del equipo, se ha pretendido llevar a cabo el trabajo desde el montaje del conjunto de datos hasta su despliegue en producción. Este objetivo ha sido alcanzado en la medida que el tiempo y los recursos disponibles han acompañado, dejar constancia de este logro es lo que se pretende en la presente memoria.

JUSTIFICACIÓN

El trabajo que se plantea es claramente justificable desde el punto de vista de la automatización de procesos. Hasta ahora, al menos que se sepa, la elaboración de informes ESG se viene haciendo de forma humana y manual, con todo lo que conlleva esto. El presente trabajo se plantea para solventar los problemas de sesgo y coste que implica cualquier proceso humano como al que nos enfrentamos. La automatización de la generación mediante técnicas avanzadas de inteligencia artificial de los criterios ESG que tan subjetivos pueden llegar a ser justifica claramente el trabajo planteado. Con la herramienta desarrollada se pretende dotar a los técnicos de unos criterios ESG cuantitativos, sin sesgo y un ahorro de costes.



4. INTRODUCCIÓN

¿QUÉ ES ESG?

Las siglas ESG, que responden en inglés, a las palabras “Environmental, Social & Governance”, en la práctica, hacen referencia a los factores que convierten a una compañía en sostenible a través de su compromiso ambiental, social y de buen gobierno, sin descuidar nunca los aspectos financieros¹.

Estas siglas están muy ligadas al concepto de inversión sostenible, un término acuñado a finales de los años 60 y que hace referencia a una filosofía de inversión ética. El progreso de la inversión sostenible ha sido tal, que en la actualidad es vital para los beneficios de las empresas cumplir con una serie de factores o parámetros sostenibles a la hora de invertir².

El origen de este acrónimo se remonta a los inicios de la década de los 2000 y ha sido el resultado de la evolución de lo que se conocía como Inversión Socialmente Responsable (ISR). Pero va más allá de lo que conocíamos como ISR, ya que tiene un enfoque holístico de todos los procesos de una compañía, permitiendo ver el alcance del impacto que trasciende al negocio.

Acertar en la identificación, gestión y medición de los criterios ESG dentro de una empresa tiene ya repercusiones directas en su capacidad para recibir inversión, en su reputación y, por extensión, en la sostenibilidad del negocio. Para las empresas es clave actuar en función de ellos, de esta manera conseguirán una mayor rentabilidad y compromiso social.

¿PARA QUÉ SIRVEN LOS CRITERIOS ESG?

Los criterios ESG tienen fronteras difusas. Lo más acertado es delimitar la capacidad de acción de la compañía en estos aspectos, de modo que los resultados intangibles sean fáciles de identificar por los inversores.

Para lograr este objetivo es crucial el asesoramiento en un índice de ESG, que permita bucear de manera más directa sobre la información de interés en materia ambiental, social y de buen gobierno de las empresas.

Ahora bien, un índice organizado y claro en materia de ESG, permitirá, en primer lugar, que los directivos y ejecutivos tomen decisiones más acertadas dentro de la compañía y, en segundo lugar, que los inversores reconozcan y premien los esfuerzos de las empresas con capital que se mantenga en el tiempo.

Para determinar qué componentes forman parte de los criterios ESG es preciso analizarlos por separado.

¹ <https://www2.deloitte.com/es/es/blog/sostenibilidad-deloitte/2021/que-son-criterios-esg-para-que-sirven.html>

² <https://www.santander.com/es/stories/que-son-los-criterios-esg-y-por-que-son-tan-relevantes>



CRITERIOS AMBIENTALES ESG

Se consideran como criterios ambientales dentro de una estrategia ESG aquellas actividades empresariales que tienen un impacto positivo en el medio ambiente. Un ejemplo de esto pueden ser las actuaciones para reducir la contaminación y la generación de residuos o la emisión de gases de efecto invernadero.

Las actividades no deben ser solo de mitigación de los efectos negativos del negocio y pueden tener una visión proactiva, como la reconversión de la matriz energética o la protección de la biodiversidad.

CRITERIOS SOCIALES ESG

En este apartado encontramos principalmente las acciones relacionadas con condiciones laborales y de respeto a los Derechos Humanos. También se incluye la gestión de relaciones con comunidades donde se opera, como población indígena, por ejemplo.

Además, este conjunto de criterios destaca por la protección y la promoción de una empresa diversa y que genere inclusión, así como un espacio saludable para los empleados y la comunidad en general.

CRITERIOS DE BUEN GOBIERNO ESG

Esta área engloba las cuestiones relacionadas con el gobierno corporativo de las organizaciones, su calidad corporativa, su cultura y sus procesos de gestión. Desde la compensación de los directivos, pasando por planes de transparencia y lucha contra las prácticas antiéticas, hasta las acertadas estrategias fiscales.

Cobra especial atención la elaboración de políticas internas sólidas y con indicadores claros que comprendan factores como la externalización, el cumplimiento normativo o la aptitud de los empleados, entre otros.

LA INVERSIÓN SOSTENIBLE Y SU CLAVE EN ESG

La inversión sostenible es aquella que suma a los criterios financieros las preocupaciones ambientales, sociales y de buen gobierno. Este modelo discrimina a las empresas por su



estrategia en materia de sostenibilidad y está creciendo de manera notable en los últimos años.³

En la siguiente ilustración se observan algunos de los criterios ESG más considerados entre las compañías consultoras ESG.

What Considerations Go Into Sustainable Investing?

Environmental	Social	Governance
Carbon emissions	Diversity & workplace policies	Board structure
Energy efficiency	Labor standards	Board composition
Water scarcity	Supply chain management	Executive compensation
Waste management	Product safety and usefulness	Political contributions & lobbying
Pollution mitigation	Customer privacy	Bribery and corruption policies & oversight
	Community impact	Strategic sustainability oversight

Examples of ESG criteria but not a complete list.

Ilustración 1. Criterios ESG [Mukut Mukherjee - <https://towardsdatascience.com/nlp-meets-sustainable-investing-d0542b3c264b>]

¿QUÉ NO ES ESG?

Ante la creciente atención acaparada por los criterios ESG en el mundo corporativo es necesario conocer qué estrategias se enmarcan dentro de sus parámetros y tienen realmente un impacto social.

La base para identificar si una estrategia está enmarcada en los criterios ESG está en los Objetivos de Desarrollo Sostenible (ODS) fijados por la ONU en 2015. Este es el marco integral para avanzar en un plan sostenible.

³ <https://www2.deloitte.com/es/es/blog/sostenibilidad-deloitte/2021/que-son-criterios-esg-para-que-sirven.html>



5. PREGUNTA DE NEGOCIO PLANTEADA

En el documento referenciado⁴ de la International Finance Corporation (IFC) y Amundi, una empresa de gestión de activos, publicado a principios de este año 2022 se afirma que:

“La inteligencia artificial y, en particular, el procesamiento del lenguaje natural (Natural Language Processing, NLP) podrían permitir a los inversores acceder y analizar datos de manera eficiente y «aumentar drásticamente» el alcance de los análisis para evaluar aspectos ambientales, sociales y de gobernanza (Environment, Social and Governance; ESG).”

Como ya se ha comentado anteriormente, los aspectos ESG adquieren una creciente importancia en la medida que también crece la sensibilidad medio ambiental de las generaciones más jóvenes y de la clase política de la mayoría de los países desarrollados. Esto genera una demanda creciente en este sentido que revoca en la actual tendencia mundial hacia la inversión sostenible

¿CUÁLES SON LOS DESAFIOS DE LOS ANÁLISIS ESG A RESOLVER?

Los desafíos a los que se enfrentan los análisis ESG son varios, de los algunos de ellos ya se han dejado entrever en las páginas anteriores. Efectivamente, uno de ellos es la falta de estandarización en las métricas que sumado a la proliferación de proveedores de calificación y de datos de ESG ha provocado un aumento de la confusión en el mercado. A esto ha contribuido que desde 2015 el fondo de datos ESG se ha triplicado a más de un billón de dólares⁵.

De manera que tanta información no estructurada y tratada de forma diferente por cada proveedor analista sólo provoca mayor confusión y frustración entre los inversores. A esta confusión se le suma la complejidad de poder comparar análisis de diferentes países o regiones que tienen legislaciones y criterios diferentes.

Otro de los grandes desafíos es eliminar el sesgo involuntario o voluntario, en tal caso considerado manipulación, de los documentos a los que se tiene acceso. Es necesario que los analistas tengan acceso a datos en bruto sin procesar ni manipular para poder llegar a extraer una idea clara y concisa del compromiso de una empresa con los criterios ESG. Este aspecto se antoja complicado cuando la mayoría de las empresas emiten sus propios informes y comunicados al respecto. Al menos hasta el momento debido a la falta de regulación al respecto. Esto provoca serias dudas en la calidad de los datos que llegan a los analistas.

Concluyendo, la pregunta a la que se busca respuesta es:

⁴ <https://research-center.amundi.com/article/artificial-intelligence-solutions-support-environmental-social-and-governance-integration-emerging>

⁵ <https://www.symanto.com/es/blog/el-papel-emergente-del-procesamiento-del-lenguaje-natural-en-los-analisis-esg>



¿ES VIABLE UNA IA BASADA EN NLP CAPAZ DE EXTRAER VALOR EN MATERIA ESG A PARTIR DE ARTÍCULOS DE NOTICIAS?

Los artículos de noticias, la información sobre proyectos facilitada a los bancos multilaterales de desarrollo (MDB), los informes de sostenibilidad y los prospectos de bonos son fuentes de datos no estructurados muy poco utilizadas. Las redes sociales y los datos de valoraciones también pueden respaldar métricas como la satisfacción de los empleados y los derechos de los trabajadores.⁶

Definitivamente el uso de técnicas de NLP se hacen imprescindibles para poder procesar tal cantidad de datos ingente en un tiempo razonable. El tiempo de elaboración de los informes es otro de los grandes retos dado que tratar tanta información por humanos se traduce en varias semanas de trabajo por expertos analistas con sus costes asociados. Además, los inversores deben utilizar la información los más a tiempo real que les sea posible.

⁶ <https://www.symanto.com/es/blog/el-papel-emergente-del-procesamiento-del-lenguaje-natural-en-los-analisis-esg>



6. ESTADO DEL ARTE DEL PROCESADO DEL LENGUAJE NATURAL PLN

El procesamiento del lenguaje natural (NLP) es el campo de estudio que se centra en la interpretación, el análisis y la manipulación de datos del lenguaje natural mediante herramientas informáticas. Las computadoras analizan, entienden y derivan significado al procesar lenguajes humanos usando NLP. Al analizar el texto, las computadoras infieren cómo hablan los humanos, y esta comprensión computarizada de los lenguajes humanos se puede explotar para numerosos casos de uso. Las aplicaciones de NLP incluyen análisis de sentimientos, donde un modelo de PNL puede predecir el tipo de sentimiento que expresa un fragmento de texto sobre el que opera, chatbots virtuales, que son robots que interactúan con humanos a través del texto, que tienen la capacidad de comprender y proporcionar lógica. respuestas a mensajes de texto enviados por humanos, reconocimiento de voz: tecnología comúnmente utilizada en convertidores de voz a texto, ahora común en teléfonos móviles y sitios web de transmisión de video, y muchos otros.

Un desafío importante en NLP radica en la propagación efectiva del conocimiento derivado o significado en una parte de los datos textuales a otra. Por ejemplo, en una oración como "*The animal did not cross the street because it was tired*" para que el modelo entienda a qué se refiere la palabra 'it' (a 'animal', no a 'street'), el significado procesado para la palabra 'animal' debe ser recordado por el modelo y recordado con precisión en el momento requerido mientras se trata de la palabra 'it'. Las herramientas de NLP han evolucionado desde las redes neuronales "vanilla" recurrentes (RNN) hasta las redes de memoria a corto plazo (LSTM), lo que lleva a las arquitecturas del modelo "Transformer" y varias variantes de las mismas, y algunos nuevos modelos revolucionarios⁷.

ESTADO DEL ARTE. TÉCNICAS NLP

Como se puede entender del párrafo anterior los Transformer son a día de hoy el estado del arte dentro de las técnicas de procesado de lenguaje natural. En las ilustraciones siguientes se puede observar la evolución del desarrollo exponencial que están experimentando los modelos Transformers año tras año, logrando multiplicar x10 el número de parámetros cada año.

Se muestran las dos gráficas para cubrir mayor espectro de modelos, dado que la primera representa más modelos pero sólo hasta el año 2020, en cambio la segunda ilustración llega hasta el 2021. Se tiene en cuenta que no aparece uno de los modelos Transformers utilizados en el trabajo, se trata de Pegasus (Google), utilizado en nuestro caso para la generación de resúmenes.

⁷ <https://zappy.ai/ai-blogs/the-current-state-of-the-art-in-natural-language-processing-nlp>

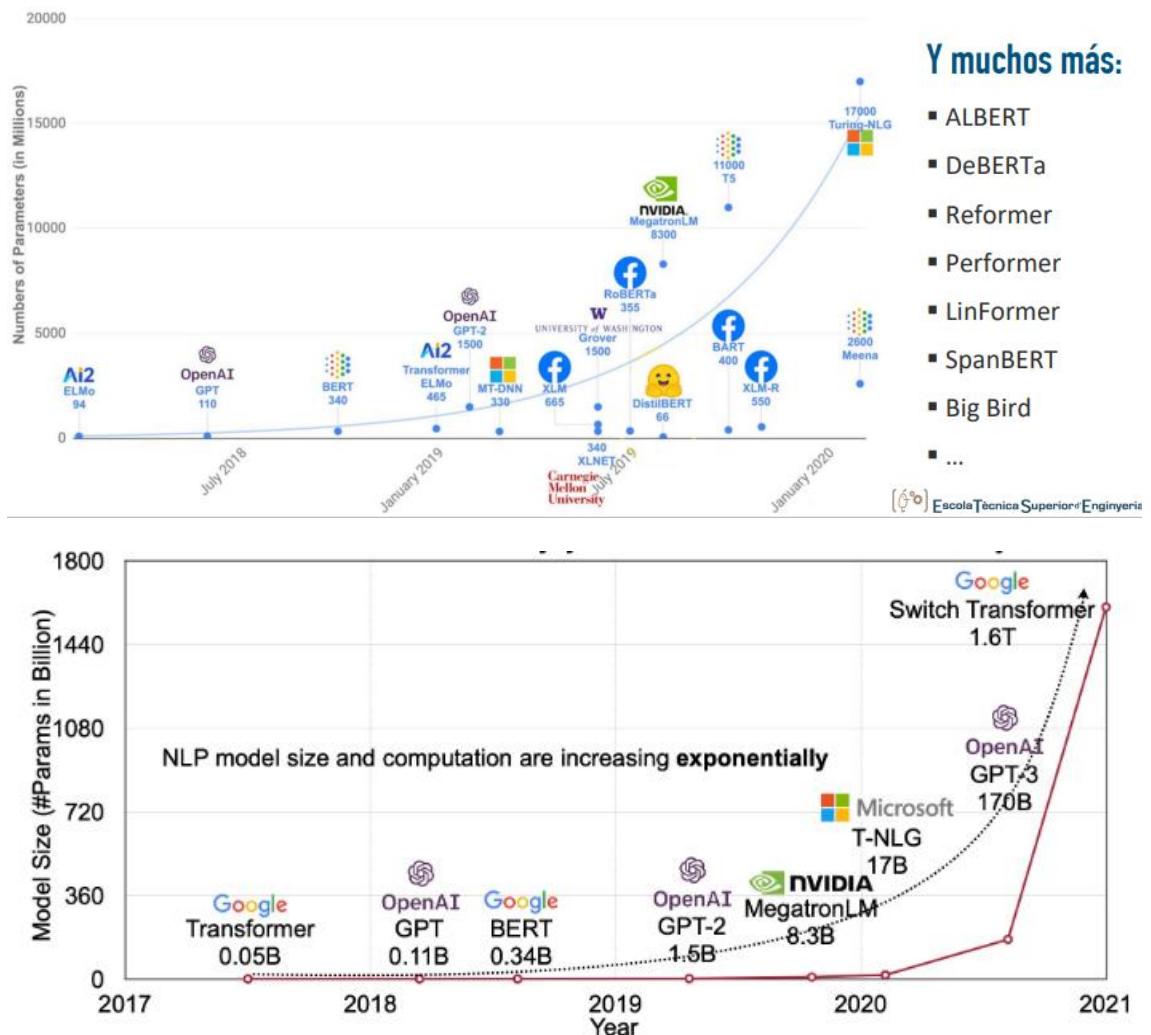


Ilustración 2. Estado del arte de los modelos Transformer NLP 20/21

[https://hanlab.mit.edu/projects/efficientnlp_old/]

En la tabla se observa la evolución histórica que están llevando a cabo los modelos Transformers, el número de parámetros ha crecido exponencialmente y esto supone que sólo las grandes compañías del sector tecnológico son las que tienen los recursos necesarios para poder llevar a cabo dicho desarrollo.

A continuación, se realizará un paseo por la historia y la evolución del transformer, para ello se hace uso de la documentación del módulo de NLP del máster, [8]

- Los primeros destellos nacen con los word embeddings (2014) [9]
- Transfer learning que sólo ocurre en la primera capa (representación de palabras con vectores).
- El 2017, investigadores del Allen Institute of AI crean ELMo [10] que extrae word embeddings contextuales. Algo que sigue en la línea del transfer learning.
- Este mismo año, en 2017, se produce uno de los mayores avances dentro de NLP, e incluso de la IA en su conjunto. En junio se publica el artículo “Attention Is All You Need”[5]



- Hay una revolución en el procesamiento del lenguaje natural, ya que proporciona resultados muy superiores a los mejores modelos del estado del arte, como las LSTMs.
 - Su arquitectura se basa puramente en atención y es de tipo encoder-decoder, que lo hace perfecto para machine translation.
 - Atención propia (entre todos los pares de tokens de la secuencia de entrada)
 - Atención entre el encoder y el decoder
- En 2018, llega el transfer learning al PLN tal cual lo conocemos hoy de la mano del modelo ULM-FiT. Una idea más parecida al transfer learning en Computer Vision.
- OpenAI publica el artículo "*Improving Language Understanding by Generative Pre-training (2018)*" donde presenta el modelo GPT.
- Utilizan solo el decoder del Transformer. El decoder es un modelo del lenguaje, por lo que permite adoptar el transfer learning. Pero es un modelo unidireccional.
 - En definitiva, OpenAI proporciona un modelo pre-entrenado basado en el transformer que permite ser ajustado en otras tareas.
 - El modelo es un stack de 12 decoders con atención propia (enmascarada).
 - Pre-entrenan el modelo con 7000 libros y lo usan para ajustarlo en tareas como clasificación de frases, similitud de frases, QA, textual entailment, etc.
- En octubre de 2018 Google introduce BERT en el artículo que se lleva su nombre "*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*"[4] Utilizan el encoder del Transformer.
- BERT resuelve el problema de la unidireccionalidad pre-entrenando un modelo del lenguaje enmascarado.
 - Además, pre-entrenan el modelo en la tarea de predicción de la siguiente frase.
 - Esto hace que el modelo sea el mejor del estado del arte en 11 tareas de PLN.
 - Durante los siguientes años se introducen variantes de BERT, como pueden ser DistillBERT, ROBERTa, ...
- A mediados de 2019 aparece el modelo "*T5 Model : Text to Text Transfer Transformer Model*", el modelo Text-to-Text-Transfer-Transformer propone reformular todas las tareas de NLP en un formato unificado de texto a texto donde la entrada y la salida son siempre cadenas de texto[11].
- Este formato hace que un modelo T5 se ajuste a múltiples tareas.
 - El modelo se entrena con el dataset C4 (Colossal Clean Crawled Corpus) de 700Gb
 - T5 logra resultados de última generación en muchos puntos de referencia de NLP y es lo suficientemente flexible como para ajustarse a una variedad de tareas posteriores importantes.
- A finales de 2019 se publica el artículo que hace referencia al modelo PEGASUS de Google dándolo a conocer en la 2020 International Conference on Machine Learning[12].



- La tarea de leer un documento y producir un resumen (por ejemplo, un resumen de un libro) para demostrar tanto la comprensión de lectura como la capacidad de escritura es una de las tareas más desafiantes en el procesamiento del lenguaje natural, lo que implica la comprensión de sentencias largas, la compresión de información y la generación de lenguaje, esto se conoce como resumen abstractivo. El paradigma dominante para entrenar modelos de aprendizaje automático para hacer esto es el aprendizaje de secuencia a secuencia (seq2seq), donde una red neuronal aprende a mapear secuencias de entrada a secuencias de salida. Si bien estos modelos seq2seq se desarrollaron inicialmente utilizando redes neuronales recurrentes (RNN), los modelos de encoder-decoder de un transformer se han visto favorecidos recientemente, ya que son más efectivos para la predicción de secuencias largas que se encuentran en el resumen.



7. DESCRIPCIÓN DEL DATASET DE PARTIDA

Para realización del presente trabajo se ha recibido como punto de partida un archivo csv, "data_as_csv.csv", que contiene un dataset con 14 columnas que a continuación se va a detallar. Cabe comentar, que inicialmente se desconocían su procedencia. El csv no aporta más información acerca de los datos contenidos. Con el proyecto ya avanzado se descubrió la procedencia del dataset, este procede de un repositorio de Github⁸. En éste tampoco aparece información de cómo se ha montado el dataset ni de dónde ha salido la información. Así que, a continuación se detalla lo que con ayuda de Alberto Oteo y horas de trabajo se ha concluido.

Como se observa en la tabla siguiente se aprecia que el dataset contiene 318332 entradas de 14 columnas cada una de ellas.

NOTA: todas las columnas no han sido utilizadas para el desarrollo del trabajo. Sólo aquellas que son necesarias y se conocía exactamente su significado.

	DATE	SourceCommonName	URL	E	S	G	Organization	Tone	PositiveTone	NegativeTone	Polarity	ActivityDensity	WordDensity	WordCount
318331	12/01/2021 21:45	ncadvertiser.com	https://www.ncadvertiser.com	FALSO	FALSO	FALSO	clorox	-3,0303E+14	1,51515E+14	4,54545E+14	6,06061E+14	2,72727E+14	0	2450
318332	12/01/2021 21:45	fox29.com	https://www.fox29.com	FALSO	FALSO	FALSO	twitter	2,86533E+14	3,72493E+14	3,4384E+14	7,16332E+14	2,34957E+14	8,59599E+14	3250
318333														

Ilustración 3. Extracto dataset original.

Columna	Descripción	dtype
DATE	Fecha y hora	datetime
SourceCommonName	Dominio web de la url de la noticia	String
URL	Dirección url de la noticia	String
E	Indica si la noticia contiene información sobre criterios E	Bool
S	Indica si la noticia contiene información sobre criterios S	Bool
G	Indica si la noticia contiene información sobre criterios G	Bool
Organization	Empresa a la que se asocia la noticia	String
Tone	Valor del tono de la noticia	Float
PositiveTone	Valor del tono positivo de la noticia	Float
NegativeTone	Valor del tono negativo de la noticia	Float
Polarity	Polaridad de la noticia	Float
ActivityDensity	NA	Float
ActivityDensity	NA	Float
WordCount	Numero de palabras de la noticia	int

Tabla 1. Detalle de las columnas del dataset original.

⁸ <https://github.com/hannahawalsh>



Sólo las columnas cuyo nombre ha quedado en negrita se ha utilizado para llevar a cabo el entrenamiento de los modelos. Las columnas que se han descartado, han sido bien, por la falta de información de cómo se han calculado o bien por desconocimiento total de su significado.

Durante el desarrollo del proyecto y concretamente, realizando el análisis y preprocesado de los datos se observó que existían datos inservibles que se tuvieron que eliminar (urls antiguas, noticias duplicadas, urls inexistentes, ...). Debido a la poda que se tuvo que realizar y considerando que los modelos de NLP requieren gran cantidad de buenos datos nos vimos en la obligación de buscar más noticias en otras fuentes para la ejecución de pruebas de test en inferencia. Estas nuevas noticias se obtuvieron del proyecto GDELT⁹. Obviamente, estos nuevos datos sólo contenían la url de la noticia.

⁹ <https://www.gdeltproject.org/data.html#rawdatafiles>

8. DESCRIPCION DEL FLUJO DE TRABAJO

Para intentar extraer valor en materia ESG de noticias, dada su URL, se ha considerado que es necesario realizar las siguientes tareas:

1. Extraer el contenido de la noticia dada su URL
2. Hacer un preprocessado y limpieza del contenido extraído
3. Opcionalmente consideramos que es posible que sea beneficioso resumir el contenido de la noticia para hacerla más asequible para las tareas siguientes por lo que se investiga esta opción
4. se realizan las siguientes tareas o bien sobre los textos resumidos o sobre el texto preprocessado, según se considere oportuno:
 - Clasificar la noticia como relevante para E,S y/o G. Se investigan los efectos de un lematizado del texto de entrada del clasificador.
 - Obtener el tono general de la noticia para poder definir la polaridad ESG
 - Extraer la empresa mas relevante de la noticia a la que aplicar el criterio ESG. Se ha limitado el alcance a empresas del SP500

La figura siguiente muestra estas tareas ordenadas de izquierda a derecha. Los cuadros de entrada/salida se muestran con línea discontinua en negro.

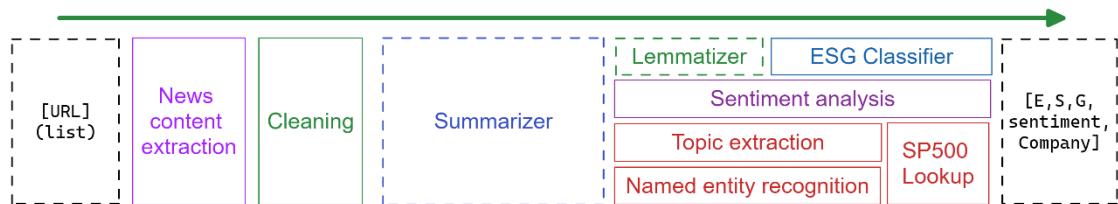


Ilustración 4. Flujo del modelo TFM CRITERIOS ESG.

En las siguientes secciones de la memoria se desarrollan las soluciones propuestas para cada una de estas tareas para finalmente concluir con una implementación del modelo global y una demostración de su puesta en producción.



9. EXTRACCIÓN DE DATOS

WEB SCRAPING

Para obtener el corpus de textos a procesar para entrenar nuestros modelos NLP, necesitamos un conjunto de URLs de webs de noticias, a las cuales aplicaremos técnicas de web-scraping para poder descargarnos principalmente el contenido, aunque también otros metadatos de estas.

Para ello se han desarrollado unos scripts de Python que acceden secuencialmente a cada una de las URLs del dataset original mediante una solicitud GET y si no hay ninguna excepción se captura el código HTML completo de la página, guardándose en un fichero HTML.

Pudimos observar que para alrededor de un 10% de las URLs, las noticias ya no estaban disponibles por lo que hubo que buscar alguna forma de recuperar esas noticias, se generó un dataset almacenando las URLs que producían excepciones y el tipo de excepción para posteriormente analizarlo e intentar extraer sus HTMLs de alguna forma alternativa.

Una vez que pudimos obtener el mayor número de ficheros de noticias (bien accediendo directamente por la URL o bien recuperándolas por <https://archive.org>) debíamos pasar a la extracción de sus contenidos.

Tras un análisis de las fuentes de las noticias, vimos que proveían de casi 8500 fuentes diferentes por lo que resultaba inviable utilizar métodos clásicos como definir patrones con Beautiful Soup [15], ya que, al proveer de cientos de webs diferentes, su estructura DOM cambiaba por completo entre unas y otras, teniendo que realizar un análisis para cada una de ellas, así como la generación de cientos de patrones para extraer sus contenidos.

Necesitábamos un enfoque más generalista y aquí entran en juego los enfoques de extracción de contenido que utilizan machine learning como se menciona en la publicación [16] *Content Extraction Using Diverse Feature Sets*, publicada en WWW en 2013.

EXTRACNET

Para esta parte se probaron varias tecnologías con unos bloques de cien noticias aleatorias, las tecnologías candidatas fueron:

- Trafilatura¹⁰.
- Dragnet¹¹.
- Extractnet¹², el cual se presentaba como una mejora de Dragnet.

¹⁰ Trafilatura - <https://trafilatura.readthedocs.io/en/latest/>

¹¹ Dragnet - <https://github.com/dragnet-org/dragnet>

¹² Extractnet - <https://github.com/currentslab/extractnet>



Finalmente, por su facilidad de uso, resultados (ya que había contenidos que no se podían extraer o resultaban en contenido vacío) y el hecho de que Extractnet extraía más campos que el resto (URL, Fecha, Autor, Título, Descripción, Titulares, Contenido...) nos decidimos por utilizar este último.

Se generó un Notebook que iba realizando las extracciones de los ficheros por bloques de 10000 ficheros, tardando una media de 5 horas en procesar cada bloque en una ejecución de un servidor Jupyter en local.

Los campos extraídos se almacenaron en un dataframe Pandas y posteriormente en ficheros csv que finalmente se unieron todos en un mismo dataset de 287786 filas.

filename	content	headlines	title	url	hostname	description	sitename	date	categories	tags	site_name	rawAuthor	authorList	rawDate	video	audio	breadcrumbs
0 42443.html	mashayab@dailynews.co.za/nP OUTICAL analysts ...	Opposition was comic and tragic... analysts sa...	Zimbabwe analysts say the deepening rifts in ...	https://www.zimbabweinstitute.com/news/opposit... zimbabweinstitute.com	zimbabweinstitute.com	POLITICAL analysts say the deepening rifts in ...	Zimbabwe Situation	2021-01-01 00:00:00			Zimbabwe	Name	[Name]	2021-01-01			
1 11269.html	Latest Survey: Prepaid Card Market Is Booming ...	Latest Survey: Prepaid Card Market Is Booming ...	Latest Survey: Prepaid Card Market Is Booming ...	https://www.opengrp.com/news/2219220/latest-sur... opengrp.com	opengrp.com	Latest Research Study on Global Prepaid Card M...	Openpr	2020-12-30 00:00:00		['press release', 'news release', 'public re...']	openPR.com	AMA research & Media	['AMA']	2020-12-30			
2 45322.html	Taylor Swift is saying goodbye to 2020 in the ...	By: 2020? With Bizarre Mid...	Taylor Swift is saying goodbye to 2020 in the ...	https://kotv.iheart.com/content/2020-12-31-tay... koton.iheart.com	koton.iheart.com	Taylor Swift is saying goodbye to 2020 in the ...	102.5 KDON	2020-12-31 00:00:00		[@Taylor Swift]	102.5 KDON	Regina Star	['Regina Star']	2020-12-31			
3 46970.html	Philip Guston, Sarasota, FL, 1967/THIS PAST ...	Free contributions consider Guston's oeuvre	THIS PAST SEPTEMBER, the directors of four ...	https://www.artforum.com/print/202101/philip-g... artforum.com	artforum.com	THIS PAST SEPTEMBER, the directors of four ...	Artforum International	2021-01-01 00:00:00		[@artforum]				2021-01-01			
4 11390.html	Louisiana Congressman-elect Luke Letlow, 41, ...	Vox Reporter Backpedals His Mocking of Republ...	Louisiana Congressman-elect Luke Letlow, 41, de...	https://townhall.com/tipsheet/juliorosas/2020/... townhall.com	Townhall.com	Louisiana Congressman-elect Luke Letlow, 41, de...	Townhall.com	2020-12-30 00:00:00		['Media Bias', 'Vox', 'coronavirus', 'covid...']	Townhall	Julio Rosas	['Julio Rosas']	2020-12-30			

Ilustración 5. Ejemplo de campos del dataset de extracción inicial



10. ANÁLISIS Y PREPROCESADO DE LOS DATOS

PRIMER ANÁLISIS: ESTRUCTURA, CONTENIDOS Y DUPLICADOS

Posterior a la obtención del dataset con los datos extraídos de sus URLs, se procedió a realizar un análisis de los campos extraídos para ver si qué campos nos serían útiles para entrenar el clasificador ESG, que a priori entrenaríamos con el contenido de cada noticia, pero que podría ser útil utilizar en algunos casos otros de los campos extraídos como el título, la descripción o los titulares o si el mismo contenido se hubiera podido extraer en cualquiera de estas otras columnas.

Esto podría suceder en los casos que, al haberse extraído con un algoritmo generalista que infiere la estructura del documento DOM del HTML, se hubiera omitido la sección del contenido principal y en su lugar se pudieran utilizar la combinación de otros de los campos en su lugar o se hubiera inferido mal la estructura del documento HTML.

Para realizar este primer análisis de cómo se habían extraído los diferentes campos, se realiza una primera limpieza sencilla de espacios en blanco y se añaden columnas de conteos de longitudes de los textos extraídos para cada una de las columnas del título, titulares, contenido y descripción. Tras realizar un análisis por casos (por ejemplo, contar casos en los que el contenido estuviera vacío y ver si se pueden usar la combinación de los otros campos, contar qué combinaciones habría... etc.), se concluyó que se utilizaría el campo '*content*' y se descartarían los ejemplos mal extraídos al no representar un alto volumen (aproximadamente un 5% de los casos).

Posteriormente se procedió a eliminar las noticias cuyo contenido fuera menor a los 500 caracteres, más tarde durante la limpieza de los textos se observaría que existían textos de longitudes muy extremas y se realizaría una poda también por el límite superior.

Después se realizó un análisis agrupando por la columna de contenido y realizando conteos, entonces nos dimos cuenta de que en el dataset original, había filas con la misma URL para algunos casos ya que se habían cruzado los datos con las empresas extraídas ('*organization*' en el dataset fuente), y en otras pese a ser diferente URL el texto del contenido era el mismo (en algunos casos el mismo texto aparecía hasta 300 veces), por lo que eliminamos la columna '*organization*' y de-duplicamos el dataset por la columna '*content*' quedando finalmente 141300 filas de las 287786 filas extraídas en el paso anterior.

Posteriormente se realizó un ajuste de párrafos en una columna adicional pues la extracción había generado espacios en blanco en lugares donde no correspondía y eran necesarios para dividir los textos en párrafos para algunas de las tareas posteriores (generación de resúmenes y '*topic modeling*').

Finalmente, para ver la distribución de noticias a nivel temporal se ha hecho un análisis agregando por fechas y tenemos datos solamente de 14 días, desde el 30 de diciembre de 2020, hasta el 12 de diciembre de 2021.

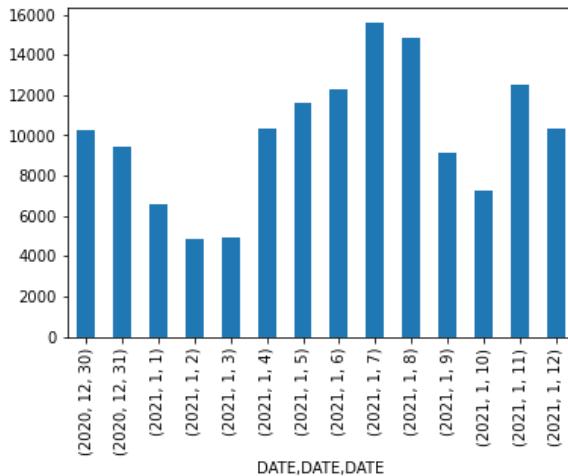


Ilustración 6. Distribución de noticias por día

SEGUNDO ANÁLISIS: BREVE ANÁLISIS DE DISTRIBUCIÓN DE LAS CLASES ESG

Se realizó un muy breve análisis en el dataset del paso anterior para ver la distribución de clases, el problema de clasificación se ha abordado como un problema multiclase, y como se pudo comprobar a continuación las clases '*E = True*' y '*G = True*' se encuentran muy desbalanceadas con respecto del resto.

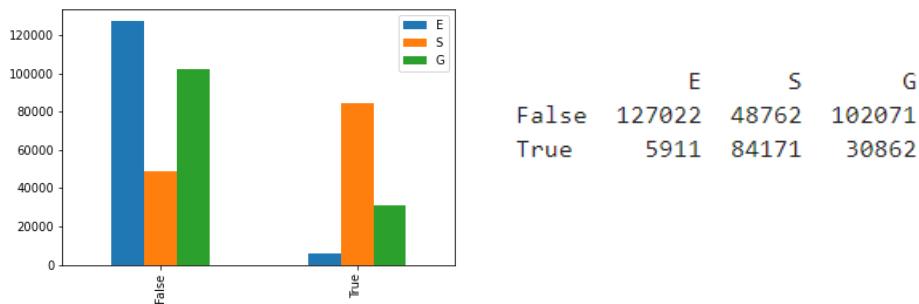


Ilustración 7. Distribución de clases

TERCER ANÁLISIS: ANÁLISIS DE LONGITUDES EN CANTIDAD DE PALABRAS

Este segundo análisis surgió después de observar un bajo rendimiento en las primeras pruebas del clasificador ESG, donde utilizamos unos modelos BERT iniciales de *TensorFlow Hub*, realizando varios intentos de *fine-tuning*, probando varios modelos y varias configuraciones de sus hiperparámetros. Estos experimentos se verán en detalle en el [apartado 15](#).

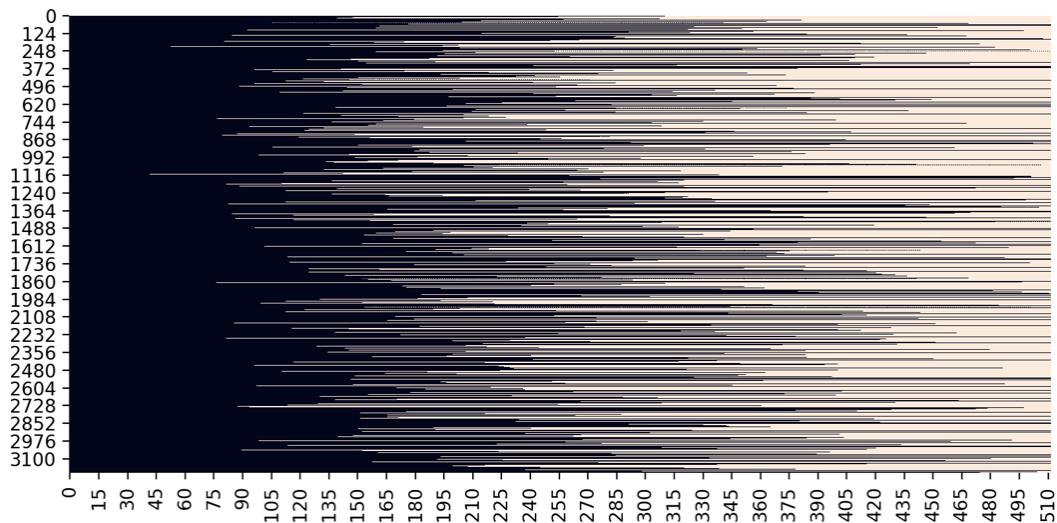


Ilustración 8. Estudio X:num_words Y:news (Batch 10%)

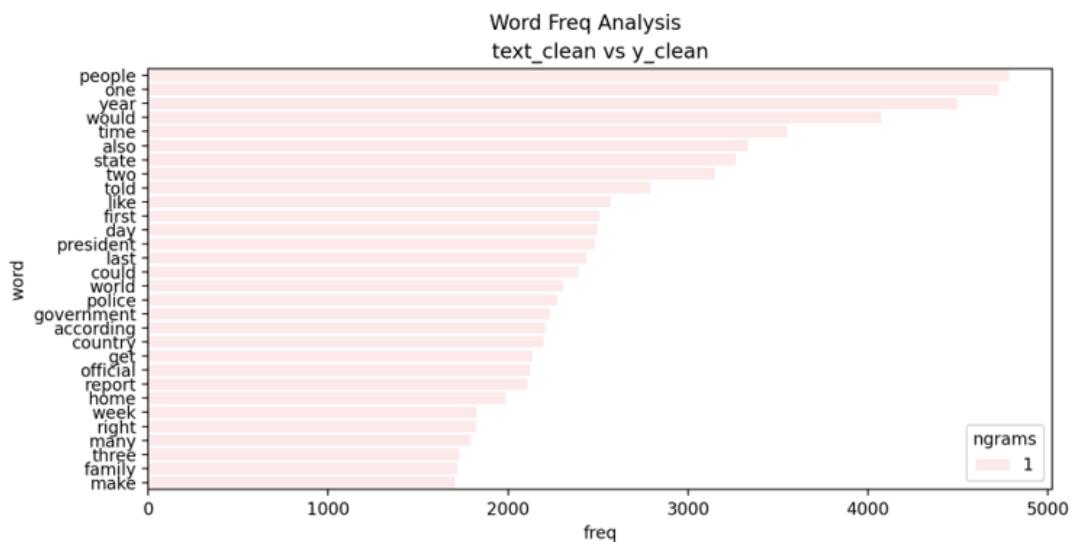


Ilustración 9. Estudio frecuencia de palabras.

Dado que se observó un desbalanceo entre clase se realizó un estudio de longitudes por clases utilizando la codificación ESG_encoded, esta codificación representa el código binario de ESG (ver tabla siguiente). Este estudio se hizo para comprobar si además del desbalanceo de clases existía un desbalanceo en la longitud de las noticias por clases. Para ello se hizo uso del gráfico de boxplot, que se puede ver a continuación.

E	S	G	ESG_Encoded
FALSE	FALSE	FALSE	0
FALSE	FALSE	TRUE	1
FALSE	TRUE	FALSE	2
FALSE	TRUE	TRUE	3
TRUE	FALSE	FALSE	4

TRUE	FALSE	TRUE	5
TRUE	TRUE	FALSE	6
TRUE	TRUE	TRUE	7

Tabla 2. ESG Encoded

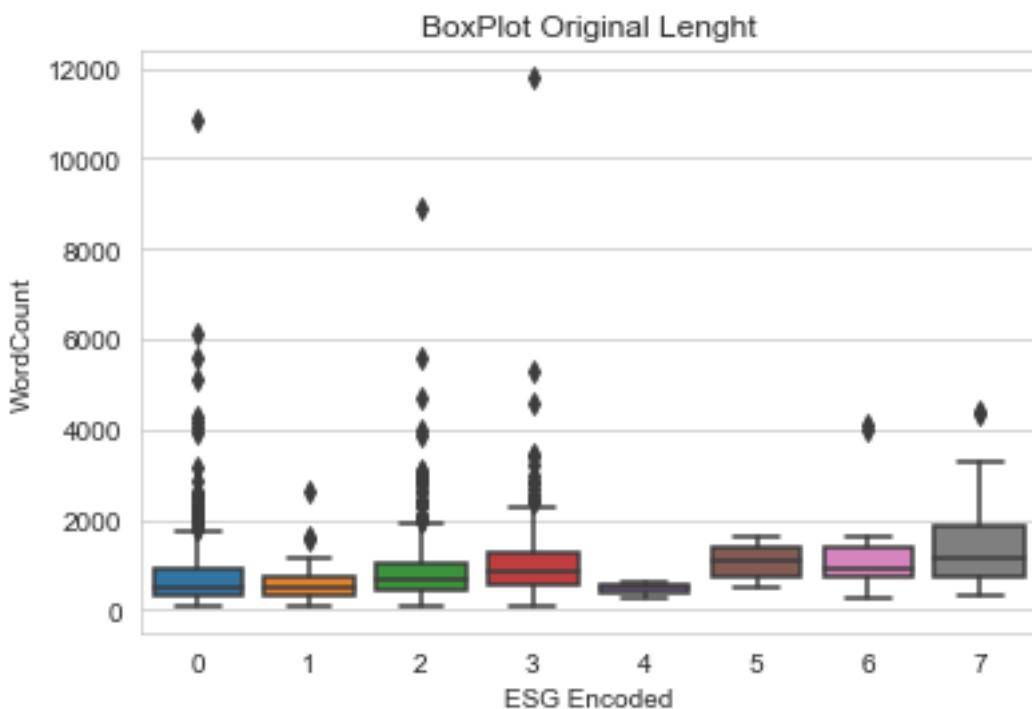


Ilustración 10. BoxPlot de longitud del texto vs ESG_Encoded.

Como se observa en la gráfica los cuartiles están bastante alineados a excepción del ESG_encoded=4, perteneciente a las noticias clasificadas como E=true, S=False y G=False. Este resultado era esperado dado que de esta clase a penas se tienen noticias, pero para el resto de noticias no se observa nada llamativo.

Adicionalmente, al darnos cuenta de que había longitudes de textos muy extremas, se optó por fijar un límite máximo de palabras por noticia, como se puede ver a continuación en la distribución de longitudes de palabras por noticia:

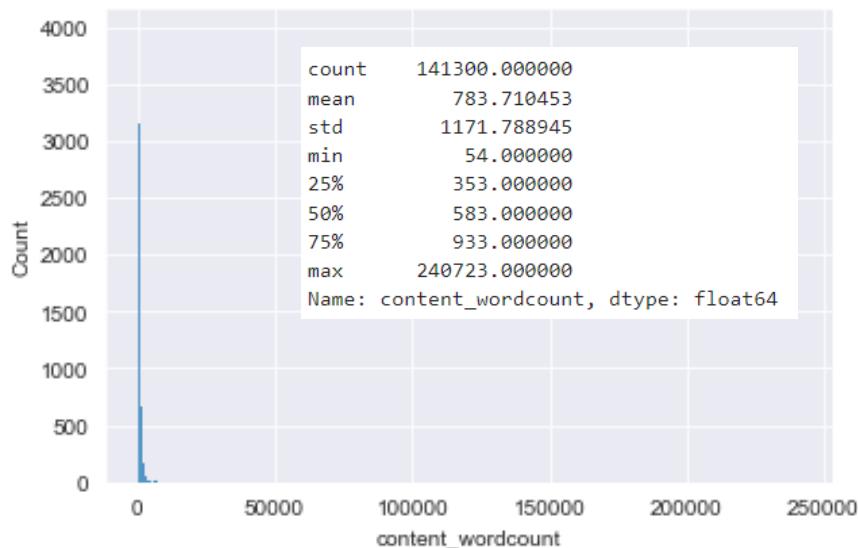


Ilustración 11.Distribución de longitudes de palabras (1)

Ampliando la gráfica y limitando el eje X de la gráfica a 5000 palabras se aprecia mejor la distribución, ver gráfica siguiente.

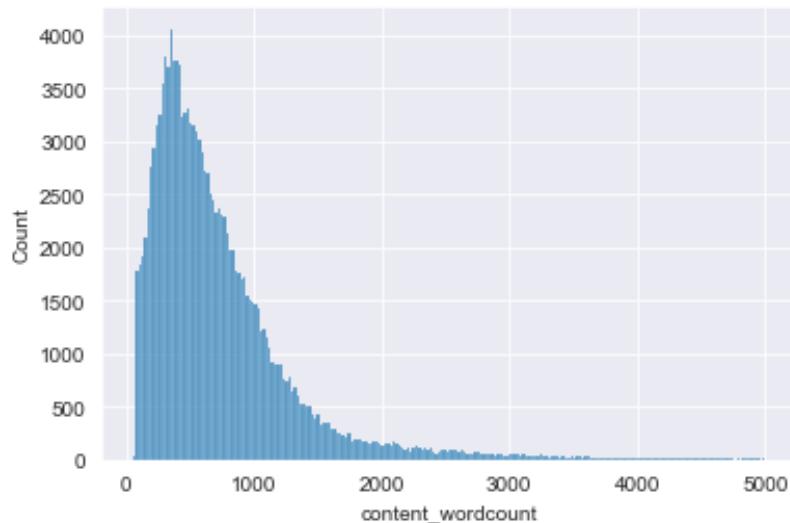


Ilustración 12.Distribución de longitudes de palabras (2)

Analizando los datos anteriores y observando el boxplot finalmente se decide fijar el límite en 4000 palabras, de esta manera no truncamos noticias de las clases menos representadas y conseguimos reducir el problema. Así y todo, el dataset se queda en 139905 ejemplos.

LIMPIEZA DE DATOS

En este punto, en el que ya se ha realizado una poda a las noticias según lo visto anteriormente se procede a realizar una limpieza de los datos no muy compleja ya que estamos trabajando con modelos Transformers y teóricamente no necesitan mucha limpieza para poder llevar la contextualización de las noticias y realizar los resúmenes. Dado que también se van a probar



otros modelos basados en LSTM se procede a generar otra columna en el dataset llamada “clean_text” con el resultado de la limpieza y otra llamada “clean&lemma” con el resultado de la limpieza y lematizado de la columna “content”. Ver apartado 9.

Para esta tarea se ha desarrollado un script de Python “cleandata_v01.py” utilizando las librerías siguientes:

- Pandas
- Spacy
- Modelo de spacy “en_core_web_sm” para la tokenización

Una vez ya lo tenemos tokenizado se procede a realizar una limpieza de los siguientes tokens:

- Signos de puntuación: .is_punct
- Stopwords: .is_stop
- Urls: .like_url
- Space: .is_space
- E-mails: .like_email
- Conjunciones: == “CONJ”

Los demás tokens también serán lematizados en este script aplicando .lemma_

En las gráficas siguientes se puede observar el efecto que produce la limpieza y el lematizado.

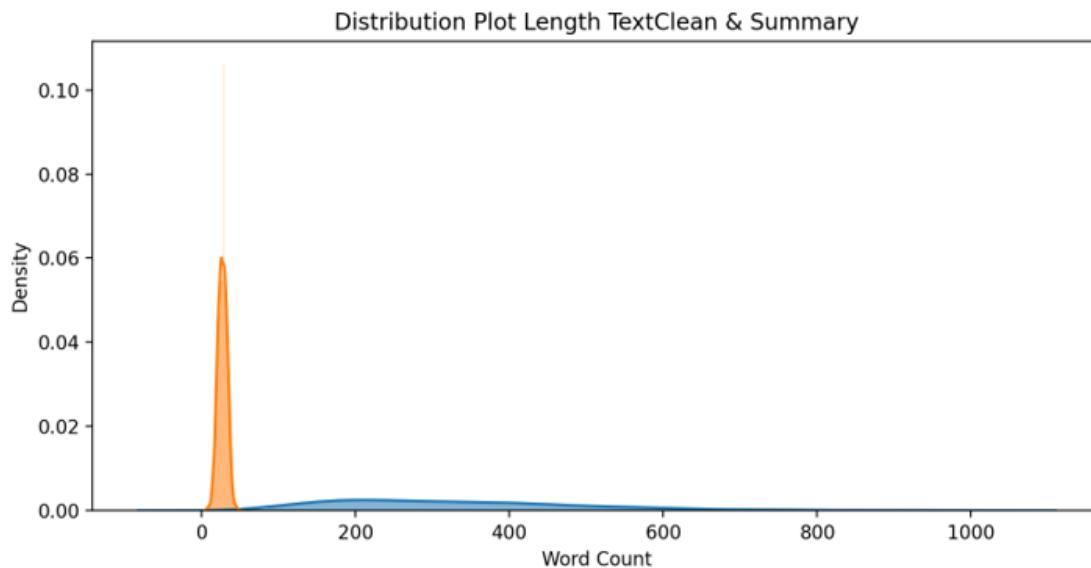


Ilustración 13. Distribución por longitud de palabras

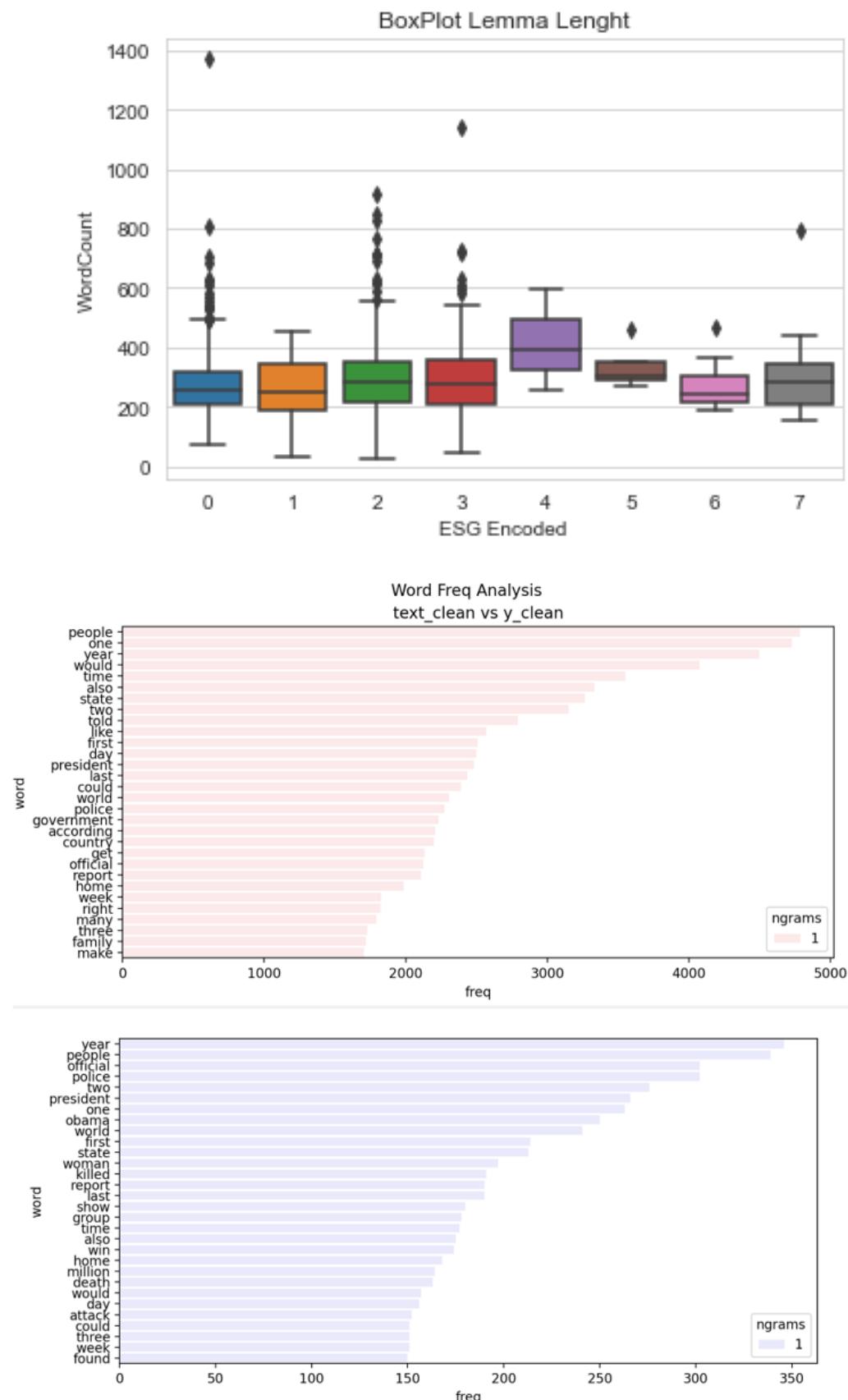


Ilustración 14. Estudio frecuencial de palabras



11. GENERACIÓN DE RESÚMENES

Tras las primeras pruebas con el clasificador BERT, y la mejora de sus resultados con las pruebas de los modelos de *Hugging Face* realizando ‘*fine-tuning*’, se ha propuesto que al existir muchas noticias cuyo contenido excede las 512 palabras, a modo de mejora, probar a utilizar el resumen de la noticia como entrada al modelo a partir de cierta longitud, con intención de comprobar si se pudiera deber a la limitación en los modelos de *Transformers* de longitud máxima de entrada de 512 tokens.

Otra posibilidad habría sido utilizar modelos de *Transformers* que permitan mayores longitudes de entrada, pero dadas nuestras limitaciones de hardware, descartamos esta opción tras investigar un poco cómo entrenar este tipo de modelos (*Longformer* [17], *Reformer* [18], *BigBird* [19]... etc.) y ver sus experiencias usando Google Colab.

Se han comparado varios modelos extractivos y abstractivos, entrenando modelos nuevos, o utilizando modelos ya pre-entrenados tanto de *TensorFlow Hub*¹³ como de *Hugging Face Hub*¹⁴, realizando para algunos de los modelos pre-entrenados, un entrenamiento adicional o ‘*fine-tuning*’ sobre el conjunto de noticias ‘*CNN/Daily Mail Corpus*’ introducido por primera vez en esta publicación 20 de 2016.

Para llevar a cabo una comparativa de los diferentes modelos de generación de automáticas de resúmenes de texto se usa la métrica Rouge [13] y se utiliza la plataforma Weight & Biases para la trazabilidad de los modelos y sus entrenamientos.

La métrica Rouge es la más empleada en las comparativas de resúmenes dado que realiza una comparativa entre la predicción por el modelo y el resumen humano basada en la similitud y solape de n-gramas entre el resumen del modelo y el ideal humano.

A continuación, se va a enumerar los diferentes modelos de generación de resúmenes con los que se han llevado a cabo pruebas, dado la cantidad de información ya incluida y que los resultados de algunos modelos no han sido buenos no nos vamos a detener en ellos para poder centrarnos en aquellos que han dado mejores resultados.

Como muestra se realiza una captura de los scripts de los modelos que peores resultados han dado, que como era de esperar no son *Transformers*.

¹³ <https://www.tensorflow.org/hub>

¹⁴ <https://huggingface.co/docs/hub/index>

	Seq2Seq (notebook).ipynb
	Seq2Seq Bi-LSTM (notebook).ipynb
	Seq2Seq_Pretrained.py
	Seq2Seq_Summary_50K.py
	Seq2Seq_Summary_Acc0.85.py
	Seq2Seq_Summary.py
	T5_TextSummarization.ipynb
	testplot.py
	TextRank (notebook).ipynb
	TextRank_Summarization.py

Tabla 3. Captura VS Code

PUROS EXTRACTIVOS

El modelo textrank [14] es el modelo base que se utiliza en todos los artículos para comparar resultados, este modelo no requiere entrenamiento se trata de un modelo basado en el modelo de grafos Google's PageRank que encuentras las sentencias más significativas en un texto.

Se han alcanzado los valores Rouge más bajos de lo esperado según su artículo, no obstante en nuestras pruebas no se ha superado 0,3 de rouge en ninguna de las métricas.

System	ROUGE score – Ngram(1,1)		
	basic (a)	stemmed (b)	stemmed no-stopwords (c)
S27	0.4814	0.5011	0.4405
S31	0.4715	0.4914	0.4160
TextRank	0.4708	0.4904	0.4229
S28	0.4703	0.4890	0.4346
S21	0.4683	0.4869	0.4222
<i>Baseline</i>	<i>0.4599</i>	<i>0.4779</i>	<i>0.4162</i>
S29	0.4502	0.4681	0.4019

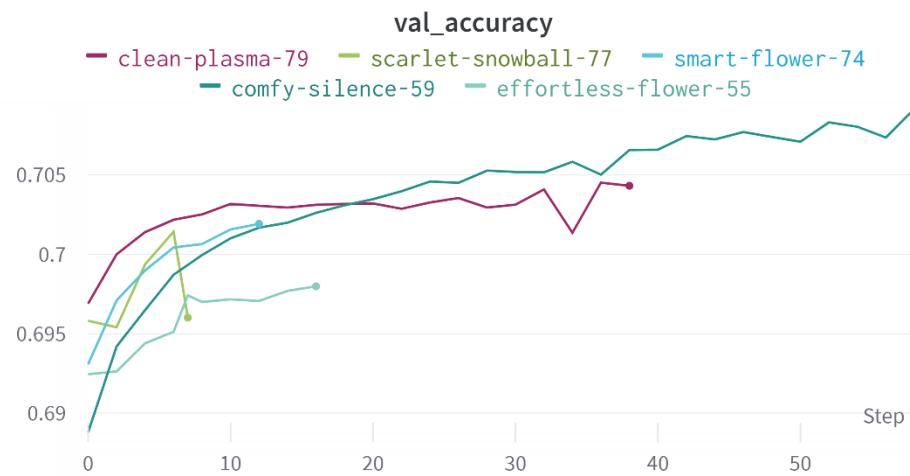
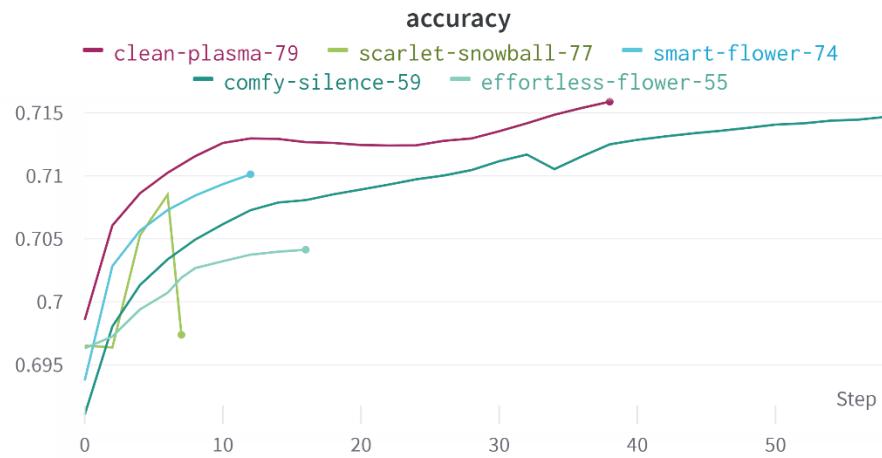
Tabla 4. Textrank Rouge score según el artículo referenciado con dataset DUC.

PUROS ABSTRACTIVOS

También se han desarrollados desde cero siguiendo el artículo de towardsdatascience¹⁵ los modelos Seq2Seq LSTM y Seq2seq LSTM Bi-directional, estos modelos basados en arquitectura seq2seq con lstm tienen el problema de pérdida de memoria y no funcionan bien con textos demasiado largas. También es cierto que sólo se ha podido realizar entrenamientos del orden de 20 a 50 épocas dependiendo de los modelos cuando se sabe que es requerido un mayor numero de épocas para alcanzar resultados notables, del orden de miles.

¹⁵ <https://towardsdatascience.com/text-summarization-with-nlp-textrank-vs-seq2seq-vs-bart-474943efeb09>

Los modelos implementados se han basado en 125 units y 250 units respectivamente, se han realizado pruebas con diferentes batch size y variando las max_length de las secuencias a entrar. A continuación, se muestran algunas de las gráficas más representativas guardadas en Weight & biases.



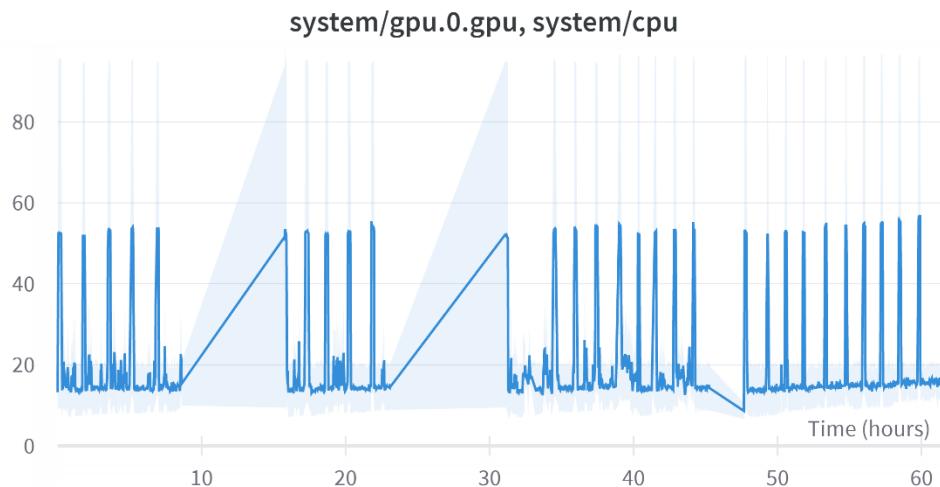


Ilustración 15. Comparativas modelos Seq2Seq LSTM & LSTM Bidireccional

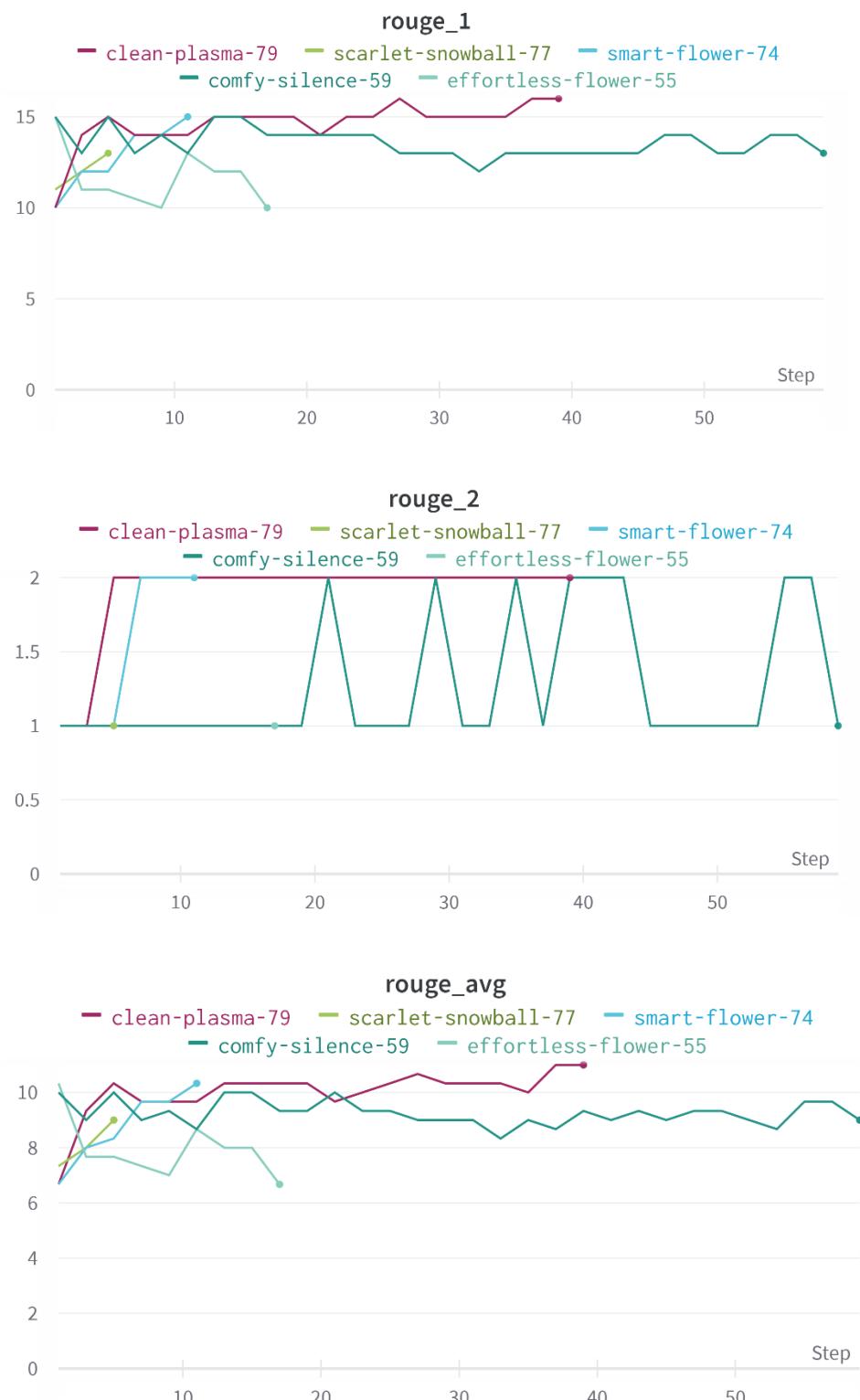


Ilustración 16. Comparativa Rouge Score de modelos Seq2seq

En las gráficas se observa dos experimentos prometedores, se trata de Smart-flower74 y clean-plasma79, estos dos experimentos son de modelos Bidireccionales. En cambio el experimento



llamado comfy-silence59 parece que se quedó estancado y no avanzaba como se aprecia en la gráfica anterior.

A continuación, se muestran los modelos de cada uno de ellos:

- Seq2seq LSTM, Units=125, BS=16, CNN-Daily, 50 epochs, loss_function="sparse categorical crossentropy".

	Name	Type	# Parameters	Output Shape
	x_in	InputLayer	0	,512
	y_in	InputLayer	0	,
	x_emb	Embedding	70638600	None, 512, 300
	y_emb	Embedding	22085700	None, None, 300
	x_lstm	LSTM	570368	,512,256,,256,,256
	y_lstm	LSTM	570368	,,256,,256,,256
	dense	TimeDistributed	18920083	None, None, 73619

Ilustración 17. Modelo Seq2seq LSTM

- Seq2seq LSTM Bidireccional, Units=250, Bs=16, CNN-Dily, 20 epochs, loss_function="sparse categorical crossentropy"

graph	Name	Type	# Parameters	Output Shape
	x_in	InputLayer	0	,512
	x_emb	Embedding	70638900	None, 512, 300
	x_lstm_1	Bidirectional	1102000	,512,500,,250,,250,,250,,250
	x_lstm_2	Bidirectional	1502000	,512,500,,250,,250,,250,,250
	x_lstm_3	Bidirectional	1502000	,,512,500,,250,,250,,250,,250
	y_in	InputLayer	0	,
	y_emb	Embedding	22086000	None, None, 300
	Concat_1	Concatenate	0	None, 500
	Concat_2	Concatenate	0	None, 500
	y_lstm	LSTM	1602000	,,500,,500,,500
	dense	TimeDistributed	36883119	None, None, 73619

Ilustración 18. Modelo Seq2seq LSTM Bidireccional

TRANSFORMERS

MODELOS PRE-ENTRENADOS (SIN FINE-TUNING)

Se han realizado pruebas de inferencia con modelos pre-entrenados tanto de BERT Base y una de sus más utilizadas DistillBERT. También se probó con T5, aunque con este sí se probó a realizar un finetuning como a continuación se podrá ver los resultados.



MODELOS PRE-ENTRENADOS (FINE-TUNING)

Se ha realizado un reentrenamiento ‘*fine-tuning*’ de dos modelos seleccionados específicamente para tareas de generación de resúmenes:

- *T5*: en su tarea específica de generación de resúmenes al que adicionalmente se le hará ‘*fine-tuning*’ sobre ‘*CNN/Daily Mail Corpus*’.
- *Pegasus*: con un preentrenamiento adicional extra ya realizado al modelo de Google, sobre el dataset ‘*Newsroom*’ [21] de 1.3 millones noticias con resúmenes abstractivos y extractivos, que se ha seleccionado para su afinado por ser el modelo con mejor puntuación Rouge en HuggingFace en aquel momento. Adicionalmente, a esta variante de *Pegasus* se le hará ‘*fine-tuning*’ sobre el mismo dataset que para las pruebas de *T5*, ‘*CNN/Daily Mail Corpus*’.

En ambos casos se ha realizado un entrenamiento ‘*fine-tuning*’ adicional sobre el dataset ‘*CNN Daily News*’ 20, se ha elegido este dataset por ser un dataset específico para resúmenes de noticias, que es el propósito final de uso del mejor de estos modelos con su configuración final.

Para la comparación de modelos se ha utilizado una misma muestra de 50000 ejemplos del dataset y se ha intentado entrenar durante 300 *steps*, no en todas las configuraciones se ha podido finalizar el entrenamiento debido a que algunas configuraciones ralentizaban mucho el entrenamiento. En la configuración de los modelos se ha variado principalmente el ‘*learning-rate*’ y se ha hecho especial hincapié en probar diferentes configuraciones del optimizador (*AdamW* [22, 23], *AdaFactor* [24] y la versión de 8-bit de *Adam*[25]) y otras técnicas¹⁶ como ‘*gradient accumulation*’, ‘*FP16 Training*’ o ‘*gradient-checkpointing*’ permitiendo jugar con el tamaño de los ‘*batches*’ de entrada a tamaños muy superiores a lo que podríamos utilizar en la plataforma utilizada para los experimentos sin estos ajustes, al utilizar mucha menos memoria pero más tiempo de ejecución.

Los experimentos se han realizado sobre *Google Colab Pro*, intentando utilizar en lo posible la GPU *Tesla T4* con 16GB de capacidad. Hay que tener en cuenta que los modelos son muy pesados y que cada ‘*epoch*’ de entrenamiento con el dataset completo duraba entre 8 y 12 horas para *T5* y más de 14 horas para *Pegasus*, habiendo descartado alguna de sus configuraciones ya que su entrenamiento para un ‘*epoch*’ completo excedía el tiempo límite ofrecido por *Colab* para una sesión.

Una vez seleccionado el mejor modelo con la mejor configuración se ha entrenado ese modelo con el dataset completo de ‘*CNN/Daily Mail Corpus*’ durante 8 epochs (cada epoch a sido una sesión de *Google Colab* por sus limitaciones de tiempo) y posteriormente se ha subido a *Huggingface Hub* para su posible posterior explotación via API.

- Pruebas sobre T5:

T5 es un modelo ‘*transformer*’ [26] que unifica varias tareas de NLP texto-a-texto, estando entre ellas la de generación de resúmenes. Para definir cada una de estas

¹⁶ <https://huggingface.co/docs/transformers/v4.19.2/en/performance>

tareas se debe utilizar un prefijo especificando la tarea seguido de los dos puntos y posteriormente indicar la secuencia o secuencias de entrada.

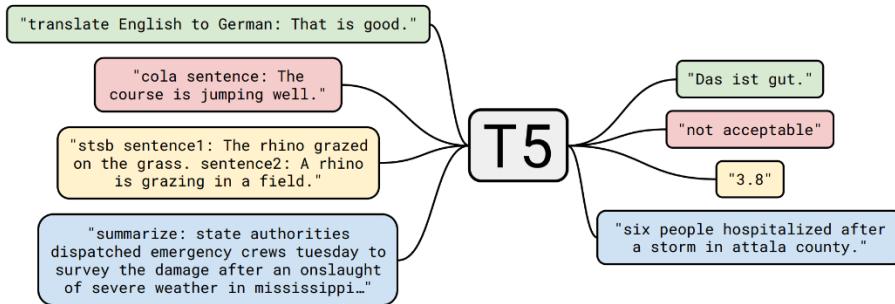


Ilustración 19.Tareas del Modelo T5

Se ha utilizado como modelo base ‘t5-base’ de Hugging Face¹⁷.

- Pruebas sobre Pegasus:
- Pegasus es un modelo ‘transformer’ específico para tareas de generación de resúmenes [27], al que se le ha pre-entrenado de forma similar a cómo se realiza el hecho de resumir, donde las oraciones importantes de un texto se eliminan o enmascaran y posteriormente se generan juntas como una secuencia de salida a partir de las oraciones restantes, similar a un resumen extractivo.

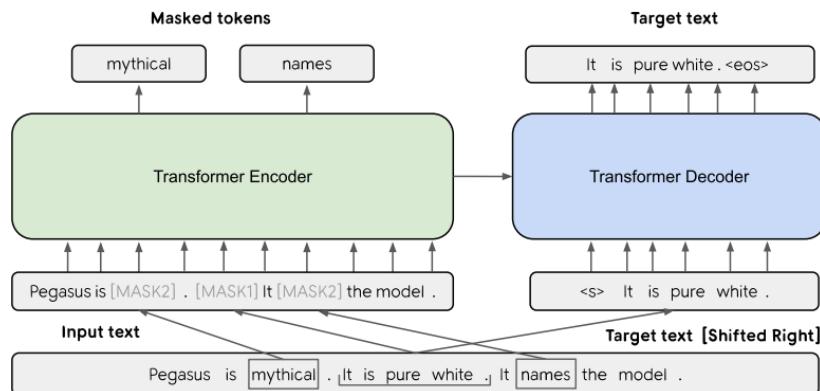


Ilustración 20. Arquitectura Pegasus

Para el entrenamiento se ha utilizado una variante de *Pegasus* de *Hugging Face* de *Newsroom*¹⁸ y para continuar con la comparación de modelos de resúmenes se ha entrenado con el mismo dataset usado en los modelos anteriores, ‘*CNN/Daily Mail Corpus*’ en su subconjunto de 50000 muestras del corpus.

¹⁷ <https://huggingface.co/t5-base>

¹⁸ <https://huggingface.co/google/pegasus-newsroom>



Pegasus es el que ha dado los mejores resultados, tras lo cual se ha entrenado el modelo durante 8 sesiones de *Colab* de una ‘epoch’ completa cada una hasta llegar a 44.2881 en puntuación Rouge1.

El modelo final se ha subido a *Hugging Face Hub*¹⁹ y se ha utilizado para comparar los resultados del clasificador de ESG frente a las noticias largas sin resumir (truncadas a los 512 tokens) y las noticias resumidas del mejor modelo extractivo.

Resultados de la selección de modelos (Rouge)

Los experimentos se han registrado en Weights & Biases²⁰ (Wandb).

Se han nombrado con la siguiente fórmula:

<Modelo>-<Optim>-<Batch Size>[-<x Gradient Acc steps>]-<Learning Rate>[-<Gradient Checkpoint>]

Se mostrarán las pruebas para T5, después las de Pegasus y finalmente el mejor de cada modelo.

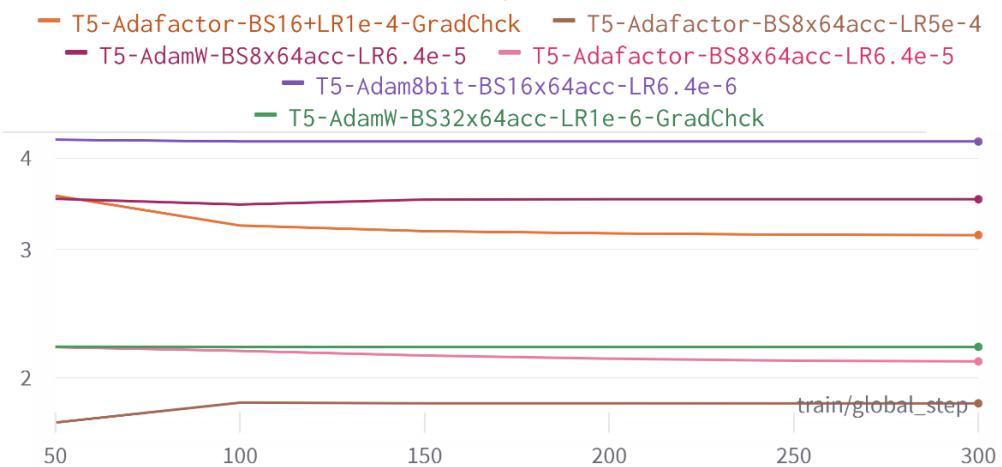
Pruebas para T5:

👁 Name (6 visualized)	Runtime	eval/rouge1 ▾	eval/rouge2	eval/rougel	eval/rougeL	eval/runtime
👁 ● T5-Adafactor-BS16+LR1e-4-GradChck	49m 17s	26.466	12.291	21.878	24.675	388.395
👁 ● T5-Adafactor-BS8x64acc-LR5e-4	3h 9m 10s	25.845	11.645	21.085	24.057	291.082
👁 ● T5-AdamW-BS8x64acc-LR6.4e-5	3h 13m 40s	25.636	11.689	21.134	23.799	333.698
👁 ● T5-Adafactor-BS8x64acc-LR6.4e-5	26m 50s	25.428	11.505	20.846	23.534	215.632
👁 ● T5-Adam8bit-BS16x64acc-LR6.4e-6	7h 11m 42s	25.171	11.315	20.583	23.251	218.921
👁 ● T5-AdamW-BS32x64acc-LR1e-6-GradChck	15h 13m 53s	24.709	10.822	20.105	22.758	698.438

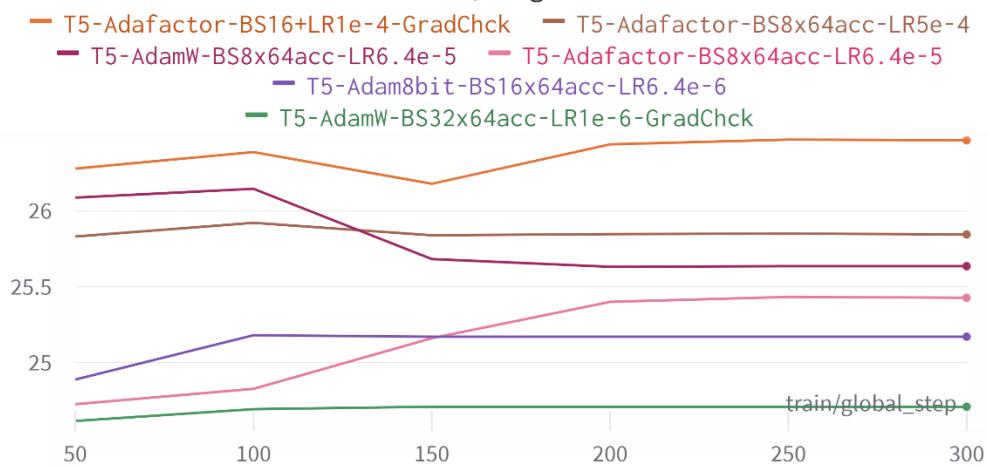
¹⁹ <https://huggingface.co/oMateos2020/pegasus-newsroom-cnn-adam8bit-bs4x64acc>

²⁰ <https://wandb.ai/site>

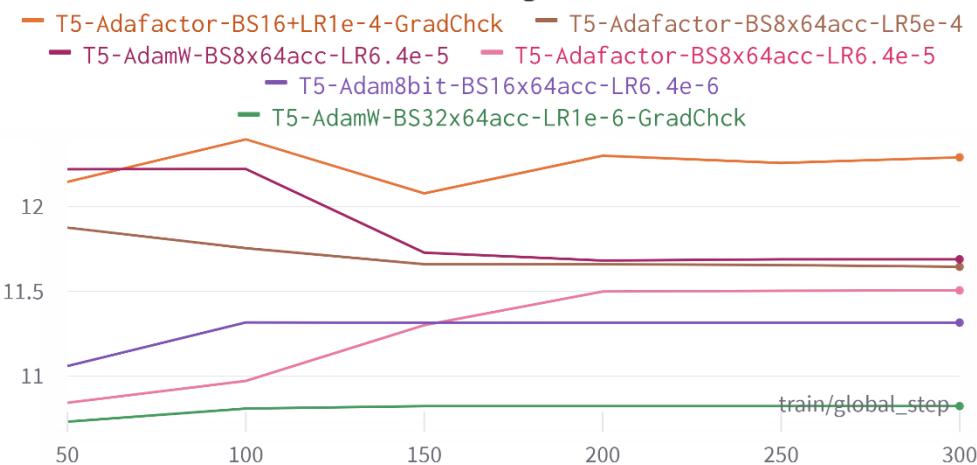
eval/loss



eval/rouge1



eval/rouge2



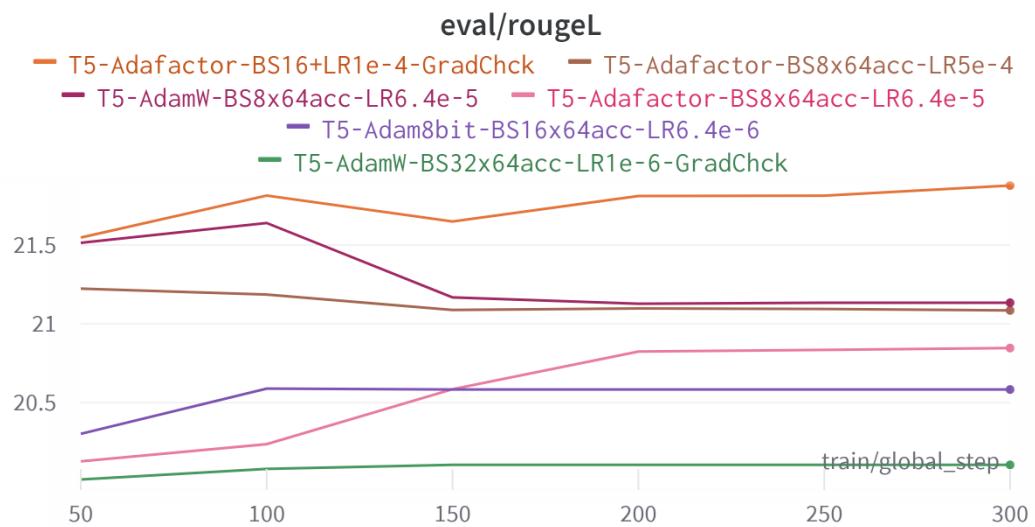


Ilustración 21. Entrenamiento T5

Pruebas para Pegasus:

Name (6 visualized)	Runtime	eval/rouge1	eval/rouge2	eval/rougel	eval/rougel	eval/runtime
Pegasus-Adam8bit-BS16x64acc-LR6.4e-5-GradChck	19h 27m 38	38.167	16.646	26.148	34.964	5963.552
Pegasus-Adam8bit-BS4x64acc-LR6.4e-5	5h 43m 43s	37.96	16.539	26.012	34.698	1096.182
Pegasus-AdamW-BS4x64acc_LR6.4e-5	6h 29m 45s	37.922	16.499	26.043	34.603	1542.648
Pegasus-Adafactor-BS4x64acc-LR6.4e-5	5h 52m 10s	37.783	16.457	25.961	34.463	1101.69
Pegasus-AdamW-BS4x64acc_LR6.4e-5-GradChck	18h 37m 7s	37.744	16.394	25.845	34.516	10685.419
Pegasus-Adafactor-BS16x64acc-LR6.4e-5-GradChck	19h 33m 18	37.428	16.196	25.706	34.098	13066.802

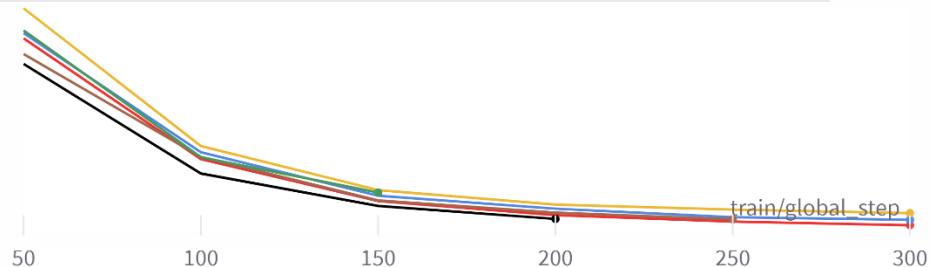
Ilustración 22. Entrenamiento Pegasus (1)

Como se puede ver a continuación, el modelo en negro ('Pegasus-Adam8bit-BS16x64acc-LR6.4e-5-GradChck') sería el mejor, pero su ejecución se ralentiza demasiado al utilizar '*gradient checkpoint*', lo que le permite subir el *batch size* del modelo de 4 a 16.

Nótese en las siguientes gráficas que debido a esta ralentización de posiblemente 4 veces la ejecución del modelo rojo no pudo completar los 300 *steps* y la sesión de *Colab* se cerró casi a las 19 horas y media de ejecución. Al estar utilizando una muestra de 50000 ejemplos del dataset original, y tras haberlo intentado con el dataset completo, que son más 250000 muestras, se ha descartado entrenar este modelo, en su lugar se entrenará el segundo mejor 'Pegasus-Adam8bit-BS4x64acc-LR6.4e-5' el modelo en color rojo.

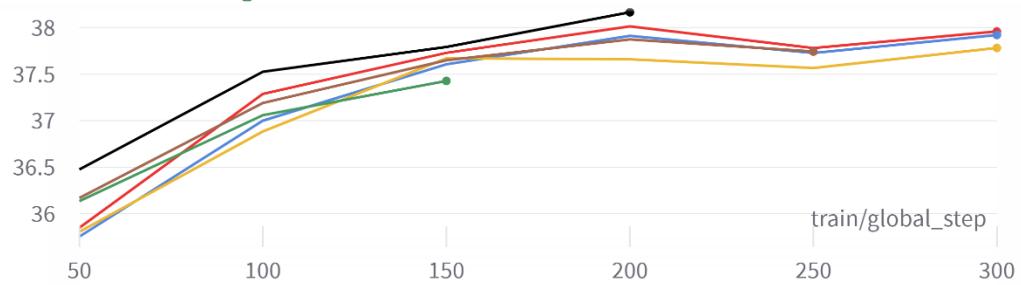
eval/loss

- Pegasus-Adam8bit-BS16x64acc-LR6.4e-5-GradChck
- Pegasus-Adam8bit-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5
- Pegasus-Adafactor-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5-GradChck
- Pegasus-Adafactor-BS16x64acc-LR6.4e-5-GradChck



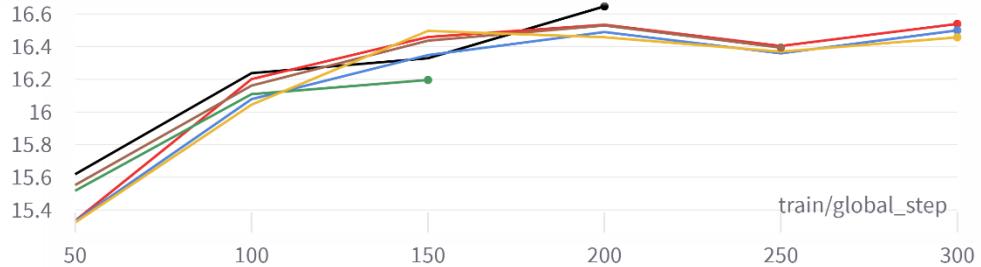
eval/rouge1

- Pegasus-Adam8bit-BS16x64acc-LR6.4e-5-GradChck
- Pegasus-Adam8bit-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5
- Pegasus-Adafactor-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5-GradChck
- Pegasus-Adafactor-BS16x64acc-LR6.4e-5-GradChck



eval/rouge2

- Pegasus-Adam8bit-BS16x64acc-LR6.4e-5-GradChck
- Pegasus-Adam8bit-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5
- Pegasus-Adafactor-BS4x64acc-LR6.4e-5
- Pegasus-AdamW-BS4x64acc_LR6.4e-5-GradChck
- Pegasus-Adafactor-BS16x64acc-LR6.4e-5-GradChck



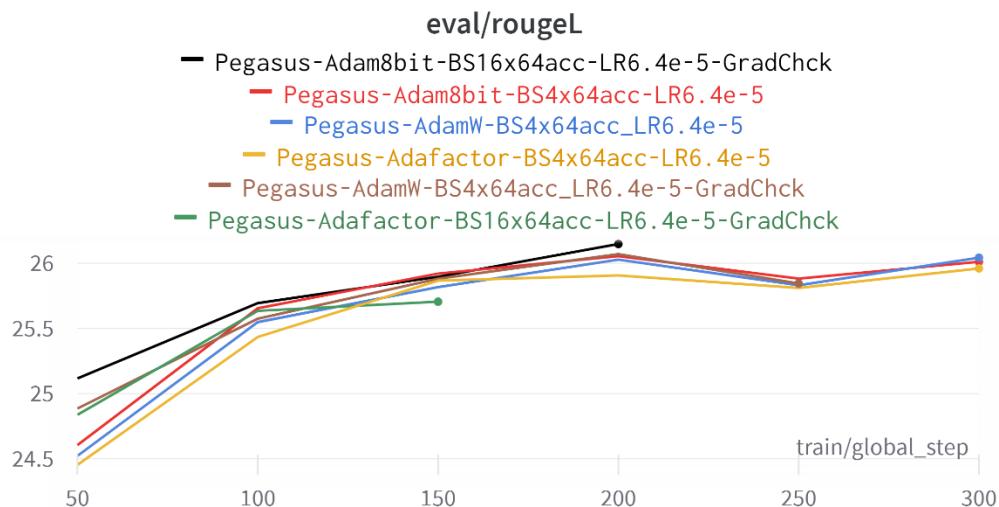


Ilustración 23. Entrenamiento Pegasus (2)

Tras seleccionar la configuración del modelo en rojo, se ejecutaron 8 sesiones de *Colab Pro* con GPU (*Tesla T4*) de 1 epoch, como las ejecuciones estaban al límite del tiempo permitido, y cada evaluación para obtener las métricas tardaba casi 4 horas, se optó por realizar solamente una evaluación al final de cada época, por lo que no se dispone de gráficas en *W&B*, pero sí de los detalles subidos a *Hugging Face*:

`pegasus-newsroom-cnn-adam8bit-bs4x64acc`

This model is a fine-tuned version of [oMateos2020/pegasus-newsroom-cnn-adam8bit-bs16x64acc](https://huggingface.co/oMateos2020/pegasus-newsroom-cnn-adam8bit-bs16x64acc) on the `cnn_dailymail` dataset. It achieves the following results on the evaluation set:

- Loss: 2.8608
- Rouge1: 44.2881
- Rouge2: 21.5487
- Rougel: 31.3798
- Rougelsum: 41.2326
- Gen Len: 71.7744

Como curiosidad hay que mencionar que el modelo se encuentra actualmente posicionado como el top1 en la categoría ‘Seq2Seq Language Modeling’ dentro de los modelos de resúmenes entrenados por la comunidad en ese dataset (‘CNN/Daily Mail Corpus’) y está mencionado en *Papers with Code*²¹ ²² desde agosto de 2022.

²¹ <https://paperswithcode.com/dataset/cnn-daily-mail-1>

²² <https://paperswithcode.com/sota/sequence-to-sequence-language-modeling-on-cnn>



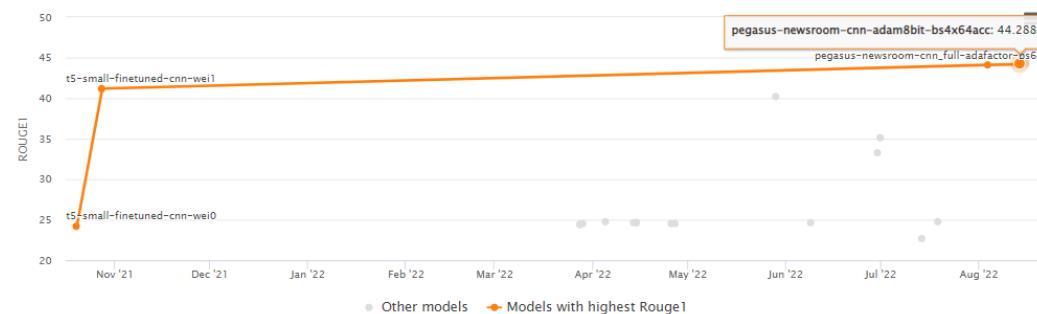
Benchmarks

[Edit](#)

Trend	Task	Dataset Variant	Best Model	Paper	Code
	Abstractive Text Summarization	CNN / Daily Mail	BRIO	paper	code
	Document Summarization	CNN / Daily Mail	PEGASUS + SummaReranker	paper	code
	Sequence-to-sequence Language Modeling	cnn_dailymail	pegasus-newsroom-cnn-adam8bit-bs4x64acc	paper	code
	Question Answering	CNN / Daily Mail	GA+MAGE	paper	code
	Text Summarization	CNN / Daily Mail (Anonymized)	HSSAS	paper	code

[Show all 12 benchmarks](#)[Sequence-to-sequence Language Modeling](#)

Sequence-to-sequence Language Modeling on cnn_dailymail

[Community Models](#)[Dataset](#)View [Rouge1](#) by [Date](#)Filter: [untagged](#)

Rank	Model	Rouge1	↑	Details	Year	Tags
1	pegasus-newsroom-cnn-adam8bit-bs4x64acc	44.288	↑	Details	2022	#ESG
2	pegasus-newsroom-cnn-adam8bit-bs4x64acc_2	44.285	↑	Details	2022	#ESG
3	pegasus-newsroom-cnn-adam8bit-bs4x64acc_3	44.268	↑	Details	2022	#ESG



12. ANÁLISIS DE SENTIMIENTO

El análisis de sentimiento es un problema de clasificación donde se busca clasificar un texto, generalmente, en tres clases; positivo, negativo y neutro.

Debido a la escasez de datos de calidad y con el objetivo de acotar el alcance del TFM se ha decidido utilizar un modelo ya existente en inferencia; es decir sin ningún tipo de modificación o fine tuning.

Se ha experimentado, sin mucho éxito, con diversos modelos los cuales están formados por un transformer BERT con capas adicionales pre-entrenados para clasificar sentimiento. Los resultados con estos modelos son pobres debido a que la mayoría están entrenados utilizando conjuntos de datos de redes sociales cuya distribución es, probablemente, muy diferente a la de nuestro conjunto de datos. Es por esto que, tras experimentar con diversos modelos, se ha decidido utilizar el modelo finbert^[7] de ProsusAI²³ el cual clasifica, de forma más razonable, las noticias con trasfondo económico.

²³ <https://huggingface.co/ProsusAI/finbert>



13. EXTRACCIÓN DE EMPRESAS

Además de la obtención de la medida ESG de cada una de las noticias, necesitamos asignar esta puntuación a las empresas para poder llegar a su valoración, que es el propósito global de este trabajo, para su posterior explotación en otras aplicaciones, como por ejemplo la generación de carteras de inversión.

Por tanto, estamos ante la necesidad adicional de tener la capacidad de extraer los nombres de las empresas que se mencionen en las noticias, por un lado, y por otro de ser capaces de seleccionar de entre todas estas, las que realmente tengan significado con respecto de lo que se está tratando en el contenido de la noticia, requerimos pues una extracción semántica de dichas entidades como nombres de empresa.

MODELO PARA EXTRAER LOS NOMBRES DE LAS EMPRESAS

Para ello rápidamente se pensó en utilizar algún modelo de reconocimiento de entidades nombradas ‘NER’ o ‘*Named Entity Recognition*’ [28] para lo cual se realizaron varias pruebas con varios modelos de la librería de NLP ‘spaCy’²⁴ en su versión 3.3.1, (modelos web del inglés ‘en_core_web_...’, en todas sus variantes incluyendo la de ‘transformers’). La idea inicial era partir de las entidades ‘ORG’ (de organizaciones) y posteriormente realizar algún tipo de filtrado para quedarnos solamente con las organizaciones que fueran empresas, ya que esta etiqueta puede incluir todo tipo de organizaciones.

Posteriormente y tras haber realizado algo de investigación se llegó a la conclusión que lo mejor sería entrenar algún modelo personalizado para que pudiera extraer una etiqueta personalizada para designar los nombres de empresas, para lo que se eligió la etiqueta ‘COMP’ de ‘company’ del inglés. La ventaja de este planteamiento es que nuestro modelo sea capaz de extraer solo este tipo de entidades teniendo en cuenta el contexto.

Para la implementación se ha utilizado ‘spaCy’ 3.4.1 y una herramienta de anotación lingüística. Inicialmente se pensó en la herramienta ‘prodigy’²⁵ de los propios desarrolladores de ‘spaCy’, pero al tener que pagar su licencia para la realización de este proyecto académico y ser su coste relativamente alto para tan solo este trabajo de fin de máster, buscamos alternativas. Finalmente nos decidimos por la herramienta ‘doccoano’²⁶, que permite realizar varios tipos de tareas de anotación y admite varios formatos de entrada y de salida, además de permitir anotaciones solapadas, algo que vimos interesante para darle flexibilidad al modelo al poder entrenarlo con entidades solapadas para reconocer nombres de empresas como, por ejemplo, que ‘JPMorgan Chase’ se pudiera reconocer tanto por ‘JPMorgan Chase’ como por ‘JPMorgan’.

El desarrollo de este modelo ha tenido varias fases: Una primera fase para la selección y anotado manual de un conjunto de 850 noticias, una siguiente fase de preprocesado de datos seguida de una selección de modelos mediante su entrenamiento y validación y para finalizar una fase de ajuste de algunos de los hiperparámetros del modelo y pruebas para tratar de

²⁴ <https://spacy.io/>

²⁵ <https://prodi.gy/>

²⁶ <https://github.com/doccano/>



mejorar los resultados de la arquitectura final, que posteriormente se ha subido a '*Hugging Face Hub*' para su posterior uso vía API.

OBTENCIÓN DE DATASETS DE ENTRENAMIENTO

Se han seleccionado de forma aleatoria 850 noticias cuyas longitudes en cantidad de palabras estuvieran entre las 185 y 450 palabras. Se han extraído en formato de texto plano exclusivamente el campo de contenido ajustado para la corrección de sus párrafos. Cada noticia en un fichero de texto diferente.

ETIQUETADO CON DOCCANO

Para esta tarea nos hemos repartido el trabajo entre los tres miembros del equipo teniendo en cuenta los mismos criterios para el anotado de los textos:

- La etiqueta sería '*COMP*', y en su definición se permite el solapamiento de etiquetas.
- Se etiquetarán solamente nombres de empresas privadas o en las que se pueda invertir o tengan financiación privada.
- Si aparece un nombre de varias formas (por ejemplo 'Walmart' en el ejemplo de abajo, se tomarán todas sus formas posibles '*Walmart*' y '*Walmart Inc*'.
- Si alguna noticia no tiene ninguna etiqueta se pasa a la siguiente sin validarla para que no se incluya en el conjunto final de entrenamiento y validación del modelo.

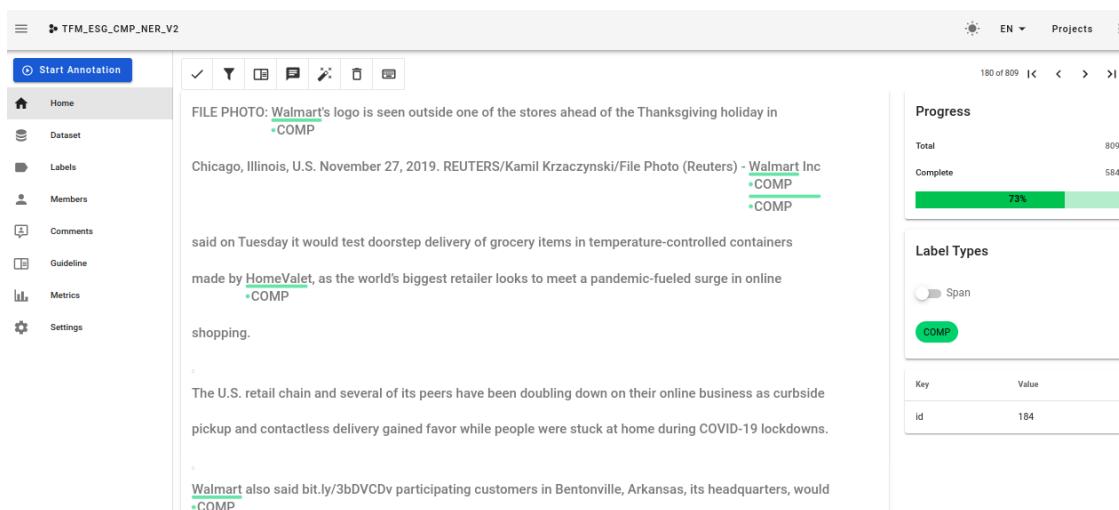


Ilustración 24. Ejemplo de anotación con doccano

Una vez finalizado el etiquetado se exportará cada bloque de textos anotados en un único fichero en formato JSONL ('*json*' con salto de línea para cada uno de los ejemplos) que es más apropiado para su preprocesado por '*spaCy*'. Se creará otro proyecto en '*doccano*' para importar los tres ficheros de anotaciones para poder exportar todas las noticias anotadas en un mismo fichero.

ENTRENAMIENTO DEL MODELO SPAN CATEGORIZER



Tras finalizar el anotado y realizar las primeras pruebas nos encontramos con el grave problema de que para *spaCy* no se pueden generar modelos NER con entidades cuyas etiquetas contengan literales que estén contenidos a su vez en otra etiqueta. Esto supuso un problema, y no queríamos echar a perder todo el trabajo de anotación manual realizado por lo que hubo que investigar si hubiera alguna forma de aprovechar todo el trabajo previo manual.

Tras profundizar en el problema en los foros de '*spaCy*' encontramos casos parecidos y afortunadamente desde la misma librería '*spaCy*' existe un tipo de modelo [29] a partir de la versión 3.1.0 llamado '*Span Categorizer*'²⁷ o '*spancat*' que precisamente sirve para la categorización de intervalos o segmentos de texto, permitiendo reconocer segmentos superpuestos y cuya implementación no varía mucho de la del código que ya habíamos desarrollado.

Para la comparación de modelos han generado varios *Notebooks* y ejecutado en *Google Colab* con GPU para el entrenamiento de todos los modelos. Para el entrenamiento de los modelos se han generado dos ficheros *jsonl* a partir del fichero que recopilaba todos los textos, uno para el entrenamiento y otro para la validación, la partición se ha realizado muestreando aleatoriamente un 80% de los datos para entrenamiento y un 20% para validación. Cada una de estas particiones se preprocesa para generar los ficheros '*spacy*' como objetos '*DocBin*'²⁸ que es el tipo de objeto serializable de *spaCy*.

El entrenamiento de los modelos se ha hecho a través de ficheros de configuración y línea de comandos y se han registrado los resultados en '*Weights & Biases*' (*Wandb*).

Se han comparado tres modelos base, dependiendo de los componentes que se incluyeran en su *pipeline*: un modelo solo con componente '*spancat*', otro con un componente adicional de '*tok2vec*'²⁹ con un modelo de "*token-to-vector*" que sirve para generar los embeddings de entrada al '*spancat*' y un tercer modelo que utiliza como entrada al *spancat* un modelo de '*transformers*'³⁰ que no es más que una componente que encapsula los transformers de '*Hugging Face*' en *spaCy*.

Resultados de las pruebas

Esta es relativa al error de la capa de Transformers del único modelo que la incluye:

²⁷ <https://spacy.io/api/spancategorizer>

²⁸ <https://spacy.io/api/docbin>

²⁹ <https://spacy.io/api/tok2vec>

³⁰ <https://spacy.io/universe/project/spacy-transformers>

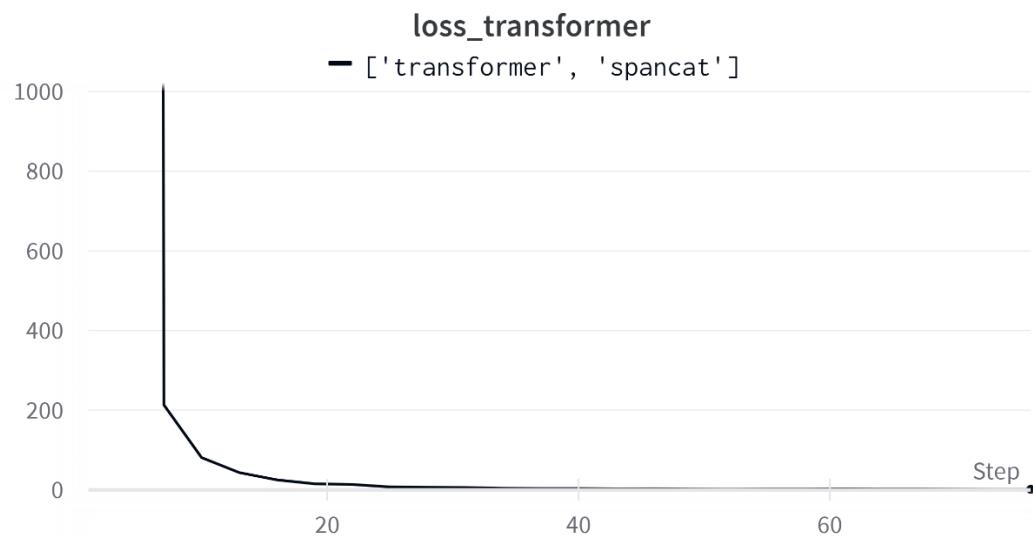


Ilustración 25. Error del transformer

Esta es relativa al error de la capa '*token-to-vectors*' del único modelo que la incluye:

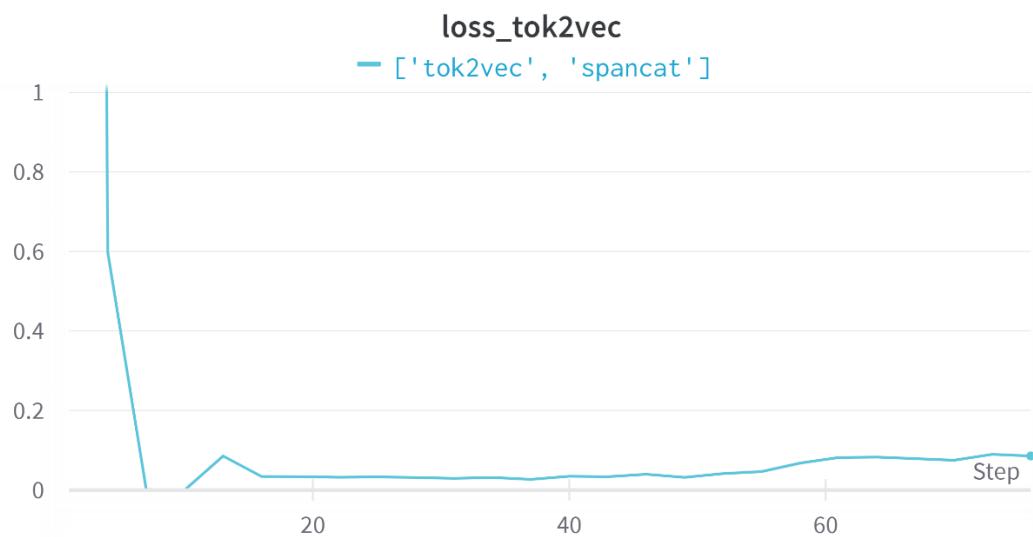


Ilustración 26. Error de tok2vec

Las siguientes ya son comunes a los 3 modelos, y se corresponden con el error de la capa '*span categorizer*' y las puntuaciones de esta para la *f1-score*, *precision* y *recall* y la última corresponde con una media de las tres:

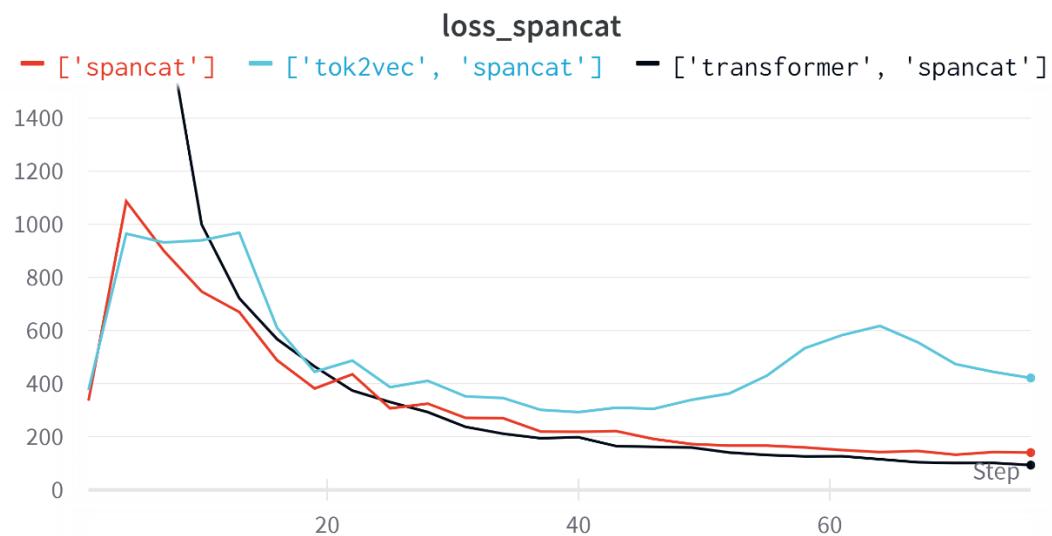


Ilustración 27. Error del spancat

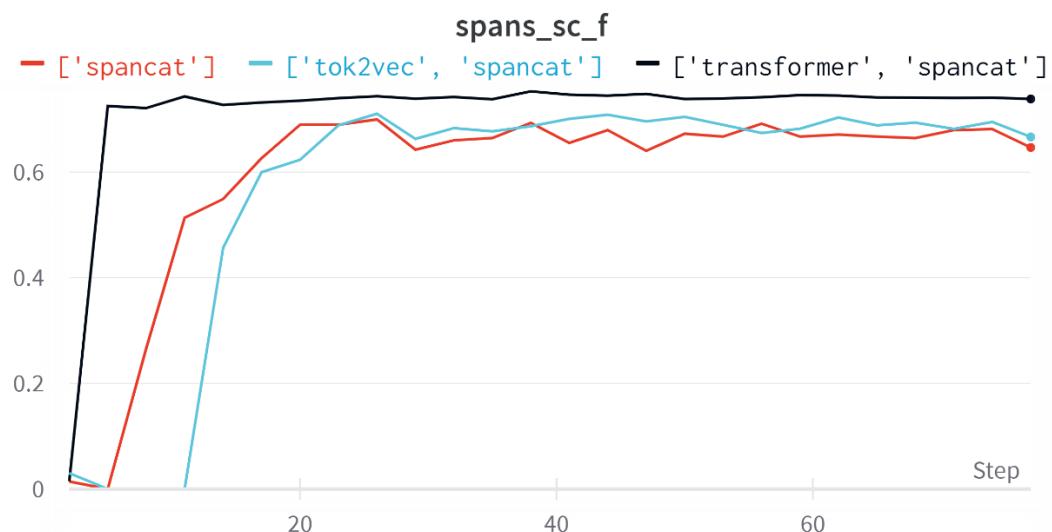
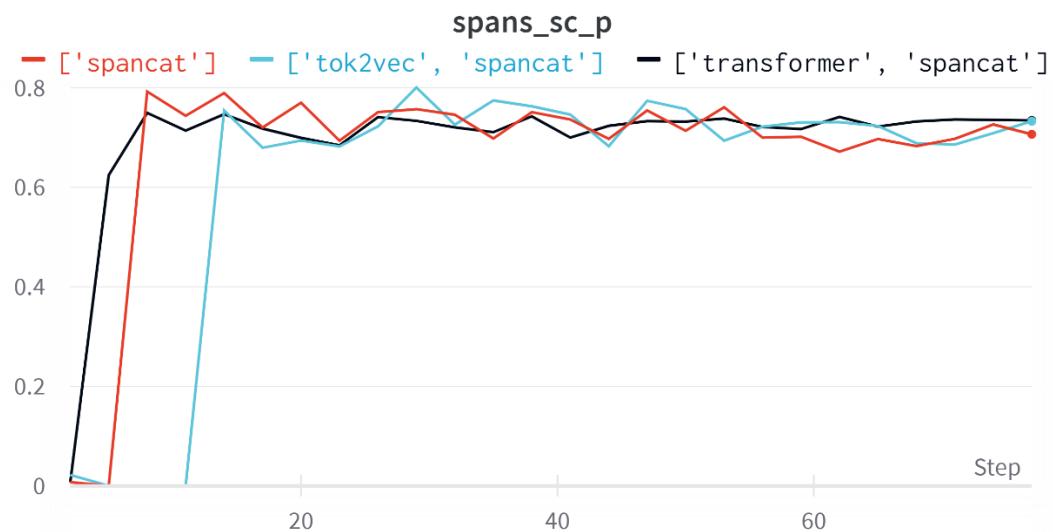
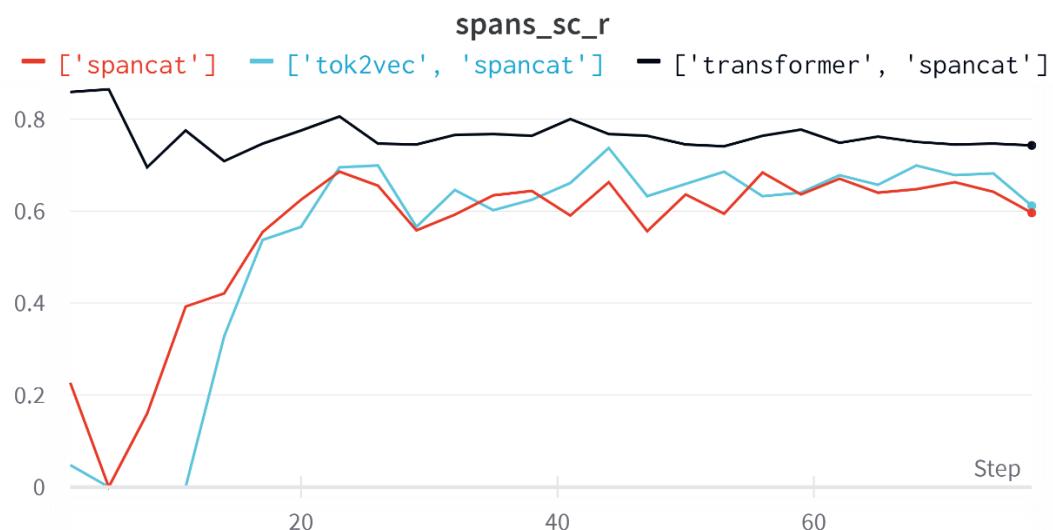


Ilustración 28. F1-score de los tres modelos

Ilustración 29. *Precision* de los tres modelosIlustración 30. *Recall* de los tres modelos

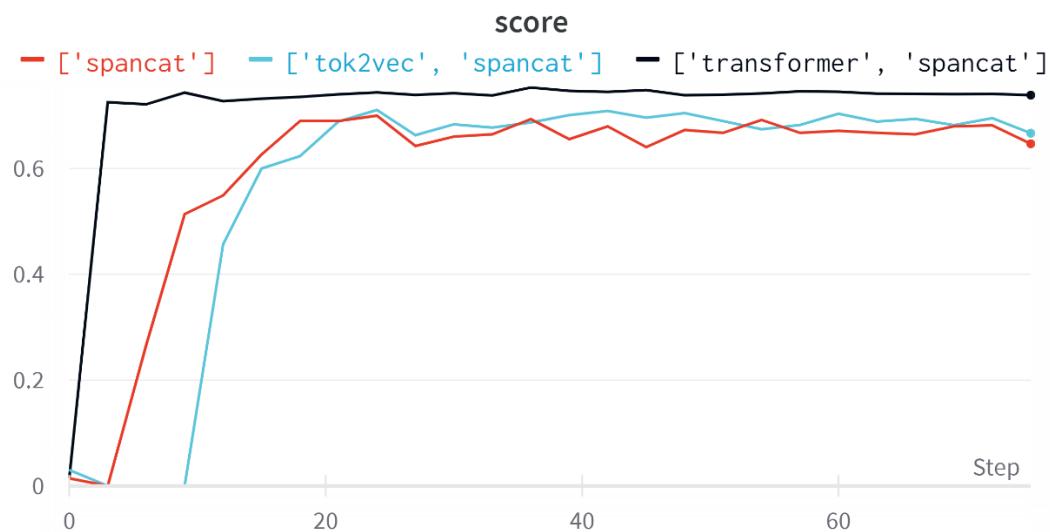


Ilustración 31. Puntuación promedio de los tres modelos

El mejor modelo como ya podíamos intuir [30] ha sido el de *transformers*, por lo que tras hacer varias pruebas en el fichero de configuración del entrenamiento se ha entrenado el que ha obtenido la mejor configuración, aunque no ha habido gran variabilidad de resultados dentro de las últimas pruebas. El mejor modelo se ha subido a '*Hugging Face*' para incorporarlo en la *API*.

Se mostrarán un ejemplo de prueba en inferencia con texto nuevo, parte de las empresas reconocidas no estaban en el dataset de anotaciones:



[] The Nintendo Switch OLED Splatoon 3 Edition is available now at multiple major retailers , including **Best Buy** ,

Walmart , and **Target** . The console just launched today , and it 's the first special - edition Switch OLED **Nintendo** has

made . If you 're interested in picking it up , you may want to order soon . It 's certainly possible that this will sell out
quickly like most other special - edition Switch consoles . The console does not come with a copy of the upcoming game
, which releases on September 9 , but it 's only \$ 10 more than the cost of the regular Switch OLED . It 's not the only
Splatoon 3 - themed product **Nintendo** is releasing to celebrate the launch of one of the biggest upcoming Nintendo

Switch exclusives . On September 9 , you 'll also be able to purchase a Splatoon 3 - themed Pro controller and carrying
case . If you still need to preorder a copy of the game , you can order a physical edition for just \$ 49 with our exclusive
promo code . Also , make sure to take a peek at our Splatoon 3 preorder guide . Multiple retailers have unique preorder
bonuses . Sadly , the free plush (arguably the best bonus) that **Walmart** was offering is currently out of stock , and it 's

unclear if more will be available at launch . But you can still get a free keychain at **Best Buy** or a sticker sheet at

GameStop . Meanwhile , **Amazon** is offering free release - day delivery for Prime members .

Ilustración 32. Ejemplo de salida de modelo de span-categorizer

BUSQUEDA FLEXIBLE MEDIANTE FUZZY SEARCH

El siguiente paso es buscar el sector de las empresas extraídas en un dataset con todas las empresas del índice S&P500³¹, y su sector asociado.

Debido a que tenemos un modelo de '*Span Categorization*' aunque el propósito es para nombrar entidades de compañías y que este es más flexible que un modelo '*NER*', puede darse el caso de querer buscar una compañía por su nombre parcial o su nombre completo (por ejemplo queremos que tanto '*Hewlett Packard Enterprise*' como '*Hewlett Packard*' o también, '*Activision*', '*Blizzard*' o '*Activision Blizzard*' se puedan encontrar en nuestro listado maestro de empresas y sectores del S&P500), entonces necesitamos algún mecanismo que nos permita esta flexibilidad para poder realizar búsquedas en el dataset por nombre con coincidencias parciales.

Aquí es donde entra en juego la búsqueda aproximada o *fuzzy search* [31]. Tras investigar qué opciones hay, se ha optado por utilizar la librería '*The Fuzz*'³² (anteriormente '*FuzzyWuzzy*'). Esta librería utiliza la distancia de Levenshtein [32] para medir dos cadenas de texto, hay dos formas de medir **[!Error! No se encuentra el origen de la referencia.]** las distancias, entre

³¹ https://es.wikipedia.org/wiki/S%26P_500

³² <https://github.com/seatgeek/thefuzz>



dos cadenas de caracteres '*Token Sort Ratio*' normaliza ambas cadenas de entrada y las ordena antes de calcular sus distancias y '*Token Set Ratio*' que realiza los pasos del anterior, pero realiza comparaciones por parejas entre las palabras de ambas cadenas para calcular las distancias. Esta segunda opción es la que nos interesa para nuestro propósito.

Se han desarrollado dos funciones que servirán para (1) buscar con flexibilidad los nombres de las mismas empresas dados por segmentos de texto de diferentes longitudes, asignando así su sector, y (2) para estandarizar el nombre de la empresa al registrado legalmente.

TOPIC MODELING PARA ASOCIACIÓN SEMÁNTICA

Todavía faltaría una pieza más si se desea que las valoraciones que calcule el modelo ESG tengan una asociación semántica con el sentido del contenido de la noticia, ya que por ejemplo se puede estar mencionando que una noticia se haya publicado en algún medio o red social y no por ello que las empresas de este medio o red social tengan sentido para que se las considere para calcular el ESG con respecto al contenido de la noticia que han publicado.

Para realizar una asociación semántica simple, ya que no tenemos etiquetas para clasificar los textos en torno a temáticas, proponemos utilizar un método no-supervisado como es *topic-modeling* para generar a partir del contenido de nuestros datos unos tópicos definidos a través de *clustering* a partir de las palabras de su contenido. De esta manera podríamos generar un mecanismo que en cierta manera pueda asociar el contenido de la noticia con determinados grupos de empresas que se mencionen en la noticia.

Una forma de realizar esto idealmente y con las piezas que tenemos, podría ser a través de los sectores de las empresas disponibles en el dataset del S&P500 utilizado en el punto anterior.

De los dos apartados anteriores podríamos extraer de la noticia las empresas, estandarizar sus nombres, restringirlas a las del S&P500 y obtener sus sectores.

Sectores

Las empresas están divididas en estos once sectores: '*Industrials*' o sector industrial, '*Health Care*' o salud, '*Information Technology*' o IT, '*Communication Services*' para comunicaciones, '*Consumer Staples*' para bienes de primera necesidad, '*Consumer Discretionary*' para bienes no esenciales, '*Utilities*' o servicios públicos, '*Financials*' para el sector financiero, '*Materials*' para la obtención de materias primas, '*Real Estate*' para el sector inmobiliario y '*Energy*' para el sector energético.

Bien, entonces lo que se pretende es mediante la generación de tópicos guiada [¡Error! No se encuentra el origen de la referencia.] a través de *BERTopic*³³, es tratar de construir un modelo que genere unos tópicos a partir de nuestros datos que definan los sectores anteriores, de esta manera se podría realizar un último filtrado de las empresas mencionadas en la noticia que realmente tengan relación con el contenido de esta.

³³ https://maartengr.github.io/BERTopic/getting_started/guided.html



Es decir, asumiendo que los tópicos estuvieran definidos de forma correcta, para una noticia nueva el modelo predice el tópico de esta noticia, por ejemplo, una noticia que hable de sanidad. Si el modelo predice que esa noticia pertenece a algún tópico de los que estén relacionados con el sector salud, habría que ir comparando el número de tópico predicho de la noticia (sanidad) con un listado de tópicos que definen cada uno de los sectores de las empresas extraídas del cuerpo de la noticia. Entonces bastaría con mantener aquellas empresas cuyo sector sea salud y filtrar las que estén alejadas de ese sector.

MODELO DE TOPIC MODELING CON BERTOPIC

Para tratar de implementar esta idea hemos utilizado BERTopic³⁴, una librería que utiliza los transformers de Hugging Face para realizar topic-modeling.

Para ello se han implementado varias funciones de limpieza del texto de la noticia específica para *topic modeling*, donde además de limpiar espacios en blanco adicionales, caracteres especiales y de puntuación, citas, correos, URLs, posesivos del inglés ('s), normalización de diacríticos, filtrado POS [¡Error! No se encuentra el origen de la referencia.] (para quedarse solamente con nombres), lematizado y *stemming*.

Además, se ha definido un '*seed topic list*' mediante una lista de palabras relacionadas³⁵ con cada uno de los sectores para tratar que los tópicos converjan hacia esos conceptos.

OBTENCIÓN DE DATASETS DE ENTRENAMIENTO

Se han extraído 1750 noticias aleatorias de cada día, de manera que tenemos 24500 noticias y de cada noticia nos quedamos solamente con el primer párrafo.

ENTRENAMIENTO DEL MODELOS TOPIC-MODELING

Tras preprocesar los textos de las 1750 noticias, se ha cargado un modelo *distilbert*³⁶ para generar los embeddings y se ha entrenado el modelo de BERTopic. Se ha limitado el tamaño de palabras mínimo a 30 y se ha fijado el número de tópicos en '*auto*' por lo que el número de tópicos final se ha reducido de 52 a 40 de forma automática:

```
Batches: 100% [██████████] 438/438 [00:15<00:00, 70.80it/s]
2022-09-12 19:59:22,010 - BERTopic - Transformed documents to Embeddings
Batches: 100% [██████████] 1/1 [00:00<00:00, 15.07it/s]
2022-09-12 19:59:44,595 - BERTopic - Reduced dimensionality
2022-09-12 19:59:45,354 - BERTopic - Clustered reduced embeddings
2022-09-12 20:00:07,119 - BERTopic - Reduced number of topics from 52 to 40
```

Ilustración 33. Ejemplo de entrenamiento de BERTopic

Visualización de los tópicos:

³⁴ <https://maartengr.github.io/BERTopic/index.html>

³⁵ Con la ayuda de <https://relatedwords.org> , <https://www.powerthesaurus.org/> y <https://www.investopedia.com>

³⁶ *distilbert-base-nli-mean-tokens*

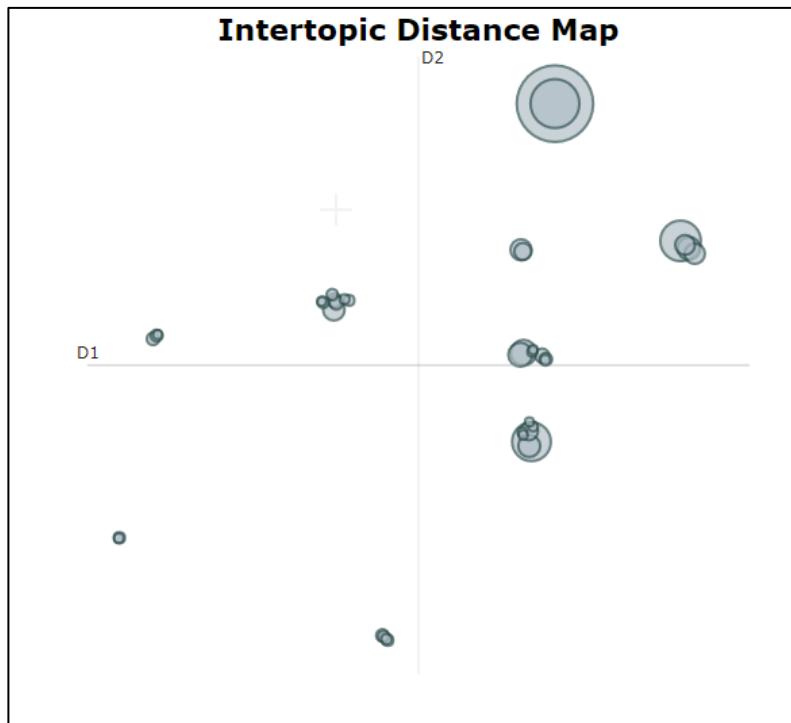


Ilustración 34. Visualización de tópicos (1)



Number of topics: 40		
Topic	Count	Name
0	-1	6468 -1_the_year_new_said
1	0	2327 0_vaccin_covid_19_covid 19
2	1	950 1_the_polic_data_death
3	2	652 2_market_global_report_the
4	3	596 3_trump_twitter_donald_donald trump
5	4	262 4_canada_canadian_vaccin_ottawa
6	5	228 5_jakarta_crash_air_sriwijaya
7	6	205 6_woman_daughter_the_actress
8	7	193 7_facebook_twitter_whatsapp_user
9	8	184 8_new_intel_devic_comput
10	9	176 9_archiv_provid_machin_contain
11	10	165 10.netflix_film_the_serি
12	11	156 11_food_market_snack_product
13	12	128 12_elect_biden_joe_joe biden
14	13	117 13_save_histori_digital_team
15	14	80 14_link_click_option_whatsapp
16	15	80 15_snow_weather_flood_road
17	16	73 16_involv_creation_content_marketwatch_news_de...
18	17	64 17_south_africa_south_africa_variant
19	18	62 18_the_association_press_association_press_as...
20	19	62 19_lavoi_season_champion_win
21	20	56 20_share_investor_bank_propos
22	21	51 21_tesla_vehicl_electr_car
23	22	51 22_subscript_pleas_premium_content_read premium...
24	23	49 23_nfl_game_quarterback_playoff
25	24	48 24_hacker_microsoft_sourc_code_code
26	25	45 25_song_music_singer_danni
27	26	44 26_feel_somalia_imag_courtesi_fear
28	27	43 27_australia_sydney_cricket_test
29	28	39 28_australia_crcc_australian_novavax

Ilustración 35. Visualización de tópicos (2)

Ejemplo de asociación:

Como ejemplo, vamos a ver como se asociaría con esta noticia, donde se predice el tema 0 que tiene que ver con el COVID y las vacunas.



```
1 text = '''UK-based pharmaceutical company AstraZeneca received approval for its COVID-19 vaccine on Wednesday (December 30).  
2 The move comes 3 weeks after the UK was the first nation to distribute the Pfizer-BioNTech vaccine.  
3 AstraZeneca's vaccine was developed in collaboration with Oxford University and is promising to health officials given  
4 that it's cheaper to make and doesn't require extreme storage temperatures, making rollout simpler.  
5  
6 Prime Minister Boris Johnson called the news of the vaccine's approval a "triumph for British science." "With two  
7 vaccines now approved, we will be able to vaccinate a greater number of people who are at highest risk, protecting  
8 them from the disease and reducing mortality and hospitalization," the UK's Department of Health said in a statement .'''  
9  
10 preds, probs = model.transform(text)  
11 print(preds, probs)  
12
```

Batches: 100% 1/1 [00:00<00:00, 17.54it/s]

2022-09-12 20:00:54,834 - BERTopic - Reduced dimensionality

2022-09-12 20:00:54,837 - BERTopic - Predicted clusters

[0] [1.]

```
1 model.get_topic(0)  
  
[('vaccin', 0.031758210147556885),  
 ('covid', 0.015450708052452742),  
 ('19', 0.01484074408441075),  
 ('covid 19', 0.014815945251043442),  
 ('health', 0.013545753695716303),  
 ('19 vaccin', 0.011515263833376727),  
 ('covid 19 vaccin', 0.0115012700791011),  
 ('coronaviru', 0.009522793346215645),  
 ('the', 0.008200589152895876),  
 ('first', 0.007537690114681161)]
```

Ilustración 36. Ejemplo de asociación semántica (1)

Supongamos que queremos comprobar las dos empresas que aparecen en el texto, *Astrazeneca* y *Pfizer*, cuyo sector sería '*Health care*', entonces mediante la función '*find_topics*' habría que generar una lista de los *topics* que más se aproximen a cada sector y comprobar si el tópico de la noticia se encuentra dentro del listado de tópicos que definen cada sector y quedarnos con la empresa para su valoración ESG si el tópico del texto está dentro de los tópicos que definen el sector de cada empresa.



```
[ ] 1 candidates = model.find_topics('Health care')
2 candidates

([0, 14, 2, 11, 17],
[0.7174093343627999,
0.6086513705343101,
0.5945532182910855,
0.5933731404100482,
0.5885149019991051])

[ ] 1 # Check if the predicted topic is within the candidate-topics from the sector keyword
2 preds[0] in candidates[0]

True

[ ] 1 candidates = model.find_topics('Financials')
2 candidates

([20, 36, 2, 16, 24],
[0.803282286089386,
0.7903151472081658,
0.7432287397518175,
0.720738270512072,
0.720065846655644])

[ ] 1 # Check if the predicted topic is within the candidate-topics from the sector keyword
2 preds[0] in candidates[0]

False

[ ] 1 candidates = model.find_topics('Information Technology')
2 candidates

([8, 36, 9, 24, 2],
[0.7764369440924759,
0.775543163149394,
```

Ilustración 37. Ejemplo de asociación semántica (2)

Que como se puede ver arriba en este caso el tópico 0 del texto estaría dentro de la lista de los tópicos de '*Health Care*' y no en los otros.

Conclusión

Como podemos ver, hay muchos tópicos que poco tienen que ver con nuestros sectores, lamentablemente no se ha realizado una selección de noticias apropiada que puedan definir bien cada uno de los sectores ya que esta parte se ha desarrollado en muy pocos días y no se ha tenido el tiempo necesario.

Además, dadas las limitaciones temporales del dataset que solo cubre 14 días incluyendo fin de año, y que muchas noticias tratan de los mismos temas, es muy difícil construir unos *topics* apropiados con este conjunto de datos, al menos para poder cubrir todos los sectores. Una posible mejora podría ser utilizar las palabras clave de cada texto para entrenar el modelo.

La idea sería incorporar esto en la API para obtener de forma inteligente un filtrado de las empresas que se mencionen en el texto de la noticia.

14. CLASIFICADOR ESG

El modelo clasificador ESG consiste en un modelo transformer preentrenado distilBERT [1] (a modo de modelo de lenguaje pre-entrenado) a cuya salida (estado oculto h final) se ha conectado una red neuronal densa para la clasificación multi-etiqueta ESG. Esto se muestra en forma de diagrama en la figura siguiente. El entrenamiento se realiza sobre el modelo completo utilizando los datos específicos de esta tarea.

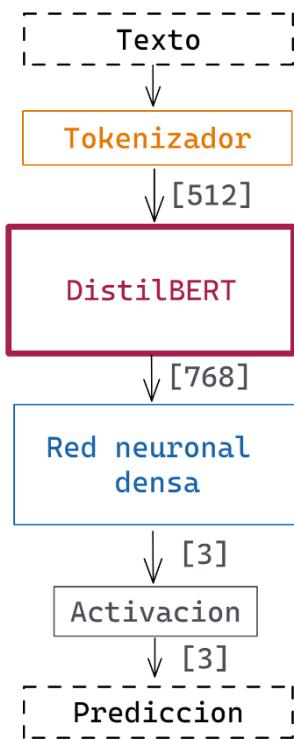


Ilustración 38 - Diagrama del modelo clasificador ESG

La metodología seguida para obtener esta configuración junto con los detalles de su implementación y entrenamiento se describen en esta sección.

IMPLEMENTACIÓN

El modelo se ha implementado en **pytorch**³⁷ utilizando las siguientes utilidades/frameworks:

- **pytorch lightning**³⁸ framework encima de pytorch que agiliza y estandariza pytorch facilitando la colaboración
- **transformers** de **huggingface**³⁹. Utilizada únicamente para acceder a los modelos de huggingface

³⁷ <https://pytorch.org/>

³⁸ <https://www.pytorchlightning.ai>

³⁹ <https://huggingface.co>



- **TensorFlow Hub**⁴⁰ es un repositorio de modelos de aprendizaje automático entrenados, listos para optimizarlos e implementarlos donde quieras.
- **Weights&biases (W&B)**⁴¹ Monitorización y almacenamiento de resultados del entrenamiento
- **Google Colab Pro**⁴² Plataforma *hosting* de Jupyter notebooks

De la implementación cabe destacar que ha sido necesario implementar un modelo personalizado para la carga de los datos y una métrica específica para clasificación multietiqueta como se comenta más adelante.

También, debido al número de parámetros del modelo, el entrenamiento solo se puede realizar (en un tiempo razonable) con un equipo con GPU dedicada por lo que ha sido necesario utilizar Google Colab Pro.

ENTRENAMIENTO E HIPERPARAMETROS

Para el entrenamiento del modelo se han utilizado tres particiones, todas ellas estratificadas para mantener la proporción original de clases de la muestra total. Se ha dividido en un 80% de los datos para entrenamiento, un 15% para test y un 5% para validación.

El entrenamiento se realiza sobre el modelo completo; es decir, se realiza un *fine-tuning* del transformer y se entrena desde cero la red densa. Al tratarse de un modelo grande es de esperar que el modelo alcance overfitting tras un bajo número de épocas debido al relativamente reducido conjunto de datos disponible. Es por esto por lo que se han implementado las siguientes medidas:

- Se ha ido reduciendo la tasa de aprendizaje gradualmente durante el entrenamiento. Esto se comenta con más detalle a continuación
- Se han implementado múltiples validaciones por época, 10, para monitorizar adecuadamente el progreso. Esta monitorización se detalla en el siguiente apartado
- Se ha implementado un *early stopping* monitorizando las pérdidas de validación con una paciencia de 10 validaciones y un umbral de 0.005.

El entrenamiento se ha realizado utilizando GPU con un batch de 24 debido a las limitaciones de memoria de la GPU. Con esta configuración, el tiempo de entrenamiento por época es de ~2.5hr con el conjunto completo de datos de entrenamiento.

Adicionalmente cabe destacar que se han implementado dos *checkpointing callbacks*; uno periódico al final de época para poder resumir entrenamiento y también un checkpointing del mejor modelo basado en las pérdidas de validación.

⁴⁰ <https://www.tensorflow.org/hub?hl=es-419>

⁴¹ <https://wandb.ai>

⁴² <https://colab.research.google.com>

MONITORIZACIÓN

Durante el entrenamiento del modelo se han ido monitorizado los siguientes parámetros para observar el proceso de entrenamiento. Estos parámetros junto con el *checkpointing* se han ido almacenando y clasificando por experimentos utilizando W&B (Weights and biases).

- Pérdidas (*binary crossentropy loss*). Calculadas utilizando un umbral de 0.5
- Matriz de confusión multietiqueta MLCM [6]
- Parámetros derivados de la matriz de confusión (*precision*, *accuracy*, *recall* y *f1*) y sus agregados (*micro*, *macro* y *weighted*)

Esta monitorización se ha realizado por cada validación (conjunto de validación) y en cada *batch* con el conjunto de entrenamiento.

Cabe destacar que la métrica MLCM no está implementada en *pytorch lightning torchmetrics* por lo que se ha tenido que implementar para este proyecto.

TASA DE APRENDIZAJE

Como primera aproximación para encontrar la tasa de aprendizaje idónea se ha utilizado la utilidad de “learning rate finder” de *pytorch lightning* la cual consiste en hacer un entrenamiento corto incrementando la tasa de aprendizaje tras cada batch y almacenando las pérdidas por cada valor de tasa de aprendizaje utilizada. El resultado es una gráfica de pérdidas frente a tasa de aprendizaje que puede usarse como orientación para encontrar la mejor tasa de aprendizaje inicial. Cabe recalcar que para tasas de aprendizaje superiores a 1e-3 se observa el efecto de “catastrophic forgetting”.

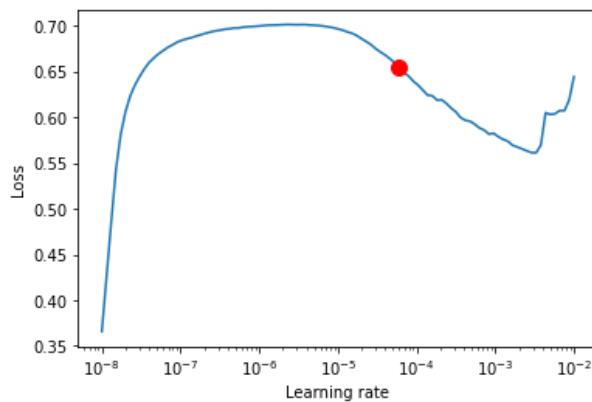


Ilustración 39 - Sugerencia de la utilidad "learning rate finder"

La tasa de aprendizaje sugerida por el algoritmo (punto rojo en la gráfica) es de 60.26e-06. Para la sugerencia el algoritmo toma el punto medio de la región con pendiente negativa mayor y constante.

Partiendo de esta tasa como referencia se han realizado una serie de entrenamientos sobre un mismo modelo para distintas tasas de aprendizaje utilizando:

- Misma semilla global
- Tasas de aprendizaje distribuidas alrededor de la sugerida por el algoritmo
- Un 10% del conjunto de datos
- Early stopping (moda resultante de 3 épocas)

Se muestran los resultados en la gráfica siguiente.

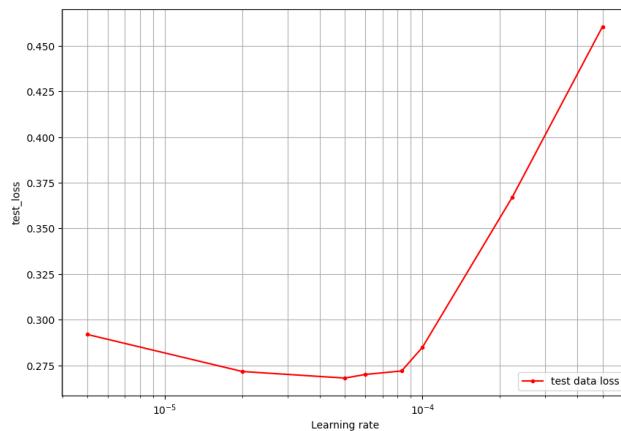


Ilustración 40 - Estudio de tasa de aprendizaje

De esta gráfica se puede observar como la tasa de aprendizaje sugerida por el algoritmo es adecuada ya que el mínimo se ha obtenido con una tasa de $50e-6$.

Por lo tanto, se ha partido de $60e-6$ como tasa inicial y esta se ha ido decreciendo durante el entrenamiento utilizando un "scheduler". El scheduler se ha seleccionado de forma empírica; obteniendo los mejores resultados con el de respuesta lineal sin "warm up" durante 5 épocas.

RED NEURONAL DENSA

La tarea de clasificación es de tipo multietiqueta; es decir las etiquetas (E, S, G) son mutuamente "no excluyentes".

Existen diversas aproximaciones para configurar una red neuronal densa para este tipo de tarea de clasificación y, empíricamente, se ha concluido que el método más directo se adapta mejor a nuestro conjunto de datos. Este método consiste en configurar la última capa densa con un número de neuronas igual al de clases (3 en este caso) utilizando a las salidas la función de activación sigmoide y empleando "binary cross-entropy" como función de perdidas.

Con esta configuración la salida de la red es de tres elementos donde cada uno representa el *grado de confianza* para cada etiqueta. Se ha tomado un umbral de 0.5 para el cálculo de las perdidas.

La configuración de la red neuronal se ha obtenido empíricamente con el objetivo de balancear el número de parámetros de la red y las pérdidas. Para ello se han realizado numerosos experimentos con distintas configuraciones de red de diferente tamaño y con las mismas condiciones iniciales (mismo conjunto de datos, semilla global y transformer).

El diagrama siguiente muestra la red que se ha observado proporciona el mejor balance pérdidas / tamaño.

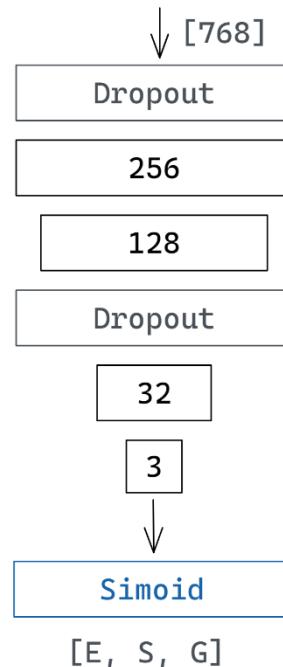


Ilustración 41 - Diagrama de la configuración final de la red neuronal densa

MODELO TRANSFORMER

Actualmente los modelos transformer [5] preentrenados, son los más utilizados en tareas de procesado del lenguaje natural al resultar mejores modelos del lenguaje que las aproximaciones "tradicionales" (conocidas como "*word embedding*" como por ejemplo *word2vec* o *GLoVe*).

Aunque hoy en día existen diversas arquitecturas de modelos transformer (GPT, BERT, T5, Switch...), la arquitectura demostrada más adecuada y usada comúnmente para tareas de clasificación es la *Bidirectional Encoder Representations from Transformers* (BERT) [4],[2]. Es por esto por lo que únicamente se han considerado transformers tipo BERT para el clasificador.

Afortunadamente estos modelos están disponibles en múltiples repositorios o "hubs" de modelos preentrenados como *huggingface*⁴³ o *tf-hub*⁴⁴. Se ha decidido utilizar *Huggingface* al tener una mayor variedad de modelos.

⁴³ <https://huggingface.co>

⁴⁴ <https://tfhub.dev>



De la multitud de transformers tipo BERT disponibles en huggingface se han considerado los siguientes para el clasificador:

- BERT base. 110M parámetros. Transformer BERT original [4]
- RoBERTa base. 124M parámetros. Optimización de BERT [3]
- DistilBERT. 66M parámetros. BERT reducido por destilación del conocimiento.[1]
- DistilRoBERTa. 82.1M parámetros. RoBERTa reducido por destilación del conocimiento.

Uno de los principales criterios considerados es el número de parámetros prefiriendo transformers *pequeños* para hacer más asequible el entrenamiento.

Para la selección del modelo se ha realizado una comparativa de modelos utilizando las mismas condiciones. Los resultados obtenidos están resumidos en la tabla siguiente.

Modelo	Accuracy	Precision
BERT-base-uncased	0.87	0.78
DistilBERT-base-uncased	0.88	0.77
RoBERTa-base	0.87	0.76
DistilRoBERTa-base	0.87	0.76

Tabla 5 - Comparativa de distintos modelos transformer

En vista a estos resultados se ha decidido utilizar distilBERT al ser el modelo de menor tamaño. Cabe destacar que por falta de tiempo⁴⁵ la tasa de aprendizaje no se ha podido optimizar para cada modelo por lo que, en teoría, se debería observar cierta mejora en los modelos grandes cuando esta se optimizase para cada uno.

PREPROCESADO Y LIMITACIÓN DE LOS DATOS DE ENTRADA

Los transformers facilitan un *tokenizador* específico por lo que en principio no es necesario ningún preprocessado del texto de entrada. Sin embargo, se ha comprobado experimentalmente que, para nuestro conjunto de datos, un lematizado simple es beneficioso. Esto se puede ver en la tabla siguiente donde se comparan los resultados de dos modelos donde la única diferencia es el lematizado aplicado al texto de entrada.

Test dataset	Raw data	Lematised
Accuracy (macro)	0.88	0.90
Precision (macro)	0.77	0.79

Tabla 6 - Comparativa entre texto lematizado y sin lematizar

⁴⁵ El tiempo de entrenamiento requerido por los modelos grandes (>3hr/época) es incompatible con la jornada laboral. Google Colab Pro no permite ejecución en segundo plano por lo que requiere de atención continuada.



Este lematizado se ha realizado utilizando el modelo de spacy “en_core_web_sm” y aplicando una pequeña limpieza a los tokens (eliminando tokens de puntuación, “stop words”, URL, correos electrónicos y conjunciones).

Los transformers basados en BERT generalmente aceptan una secuencia máxima de 512 tokens teniendo dos tokens reservados; **[CLS]** el primer token de la secuencia y **[SEP]** para concluir o separar segmentos dentro de la secuencia. Los textos de entrada fácilmente pueden exceder este límite por lo que se han considerado las siguientes opciones para tratar los textos que exceden los 510 tokens:

- Truncado
- Método jerárquico donde el texto se divide en bloques de 510 tokens que se pasan al transformador y cuyos resultados se agregan combinando las representaciones del texto entero
- Reducción de la longitud del texto utilizando otros modelos transformer diferentes

TRUNCADO

Una de las opciones más sencillas es un truncado simple. Es decir, considerar únicamente 510⁴⁶ tokens descartando el resto. Se han probado distintas configuraciones de truncado.

- La denominada "head-only" consiste en tomar los primeros 512 tokens
- "tail-only" donde se toman los últimos 510 tokens
- y "head+tail", utilizando una ratio de primeros/últimos tokens (0.25/0.75 como se sugiere en [2])

Los resultados de esta comparativa se muestran en la tabla siguiente:

Test dataset	"head-only"	"head+tail" 25%/75%	"tail-only"
Accuracy (macro)	0.88	0.87	0.87
Precision (macro)	0.77	0.76	0.75

Tabla 7 - Comparativa de distintos métodos de truncado

Se observa que en nuestro caso se obtienen mejores resultados truncando a *head-only*.

MÉTODO JERÁRQUICO

El método jerárquico no se ha conseguido implementar en el lazo de entrenamiento de forma eficiente por lo que ha terminado extendiendo el tiempo de entrenamiento a niveles inaceptables. No obstante, se han realizado una serie de pruebas manuales en las que los resultados eran muy similares e incluso inferiores a los obtenidos con truncado simple. Por lo tanto, se ha decidido descartar este método al no observarse una clara mejora que justificase el tiempo requerido para su implementación de forma eficiente.

REDUCCIÓN UTILIZANDO TRANSFORMER

⁴⁶ Primer y último tokens son **[CLS]** y **[SEP]**



Este método consiste en utilizar un transformer para resumir el texto reduciéndolo a un tamaño de aproximadamente 510 palabras. Se recurre a truncado de tipo *head-only* cuando el texto resumido excede los 510 tokens. Los modelos utilizados para este método de reducción se han descrito en la sección 11.

Los resultados obtenidos se muestran en la tabla siguiente.

Model	Accuracy (macro)	Precision (macro)
Pegasus	0.89	0.76
DistilBERT	0.91	0.79
Sum_by_paragraph	0.90	0.80
Sum_by_paragraph lemmatised	0.91	0.80
Sum_by_paragraph512	0.90	0.81
Truncated	0.88	0.77
Truncated Lemmatised	0.93	0.83

Tabla 8 - Comparativa de distintos métodos de generación de resumen

Se aprecia como sí que suponen una mejora respecto a un truncado simple pero no consiguen superar a los resultados obtenidos con un lematizado y truncado del texto.

RESULTADOS

La configuración final del modelo es la siguiente:

- El texto se preprocesa con un lematizado simple antes de pasarlo al tokenizador
- Se aplica un truncado de tipo *head-only* a la salida del tokenizador
- El modelo se entrena con un *batch* de 24 y una tasa de aprendizaje de 60e-6 la cual se va decrementando gradualmente de forma lineal hasta cero durante 5 épocas

El mejor modelo se obtiene en la tercera época con unas pérdidas de validación de 0.161 y 0.160 de test. La siguiente tabla muestra los resultados obtenidos por etiqueta.

	E	S	G	Sin etiquetar	Macro
Accuracy	0.98	0.91	0.93	0.89	0.93
Precision	0.77	0.93	0.87	0.77	0.83

Tabla 9 - Mejores resultados del modelo clasificador ESG

Y la matriz de confusión se muestra en la gráfica siguiente.

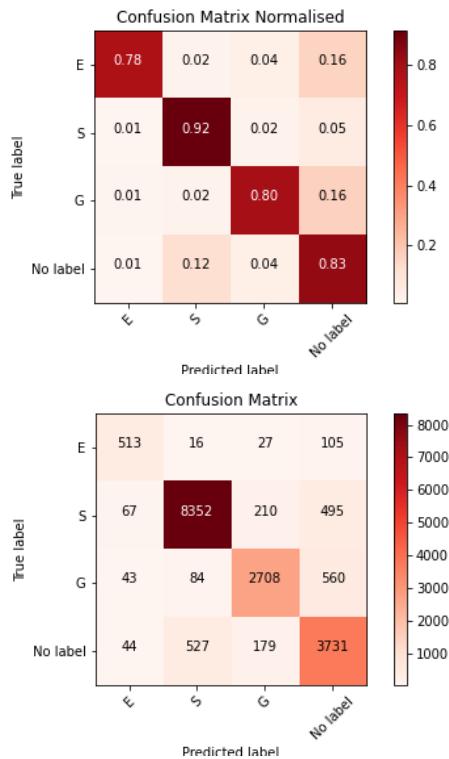


Ilustración 42 - Matriz de confusión y su versión normalizada obtenida con el mejor modelo clasificador sobre el conjunto de test

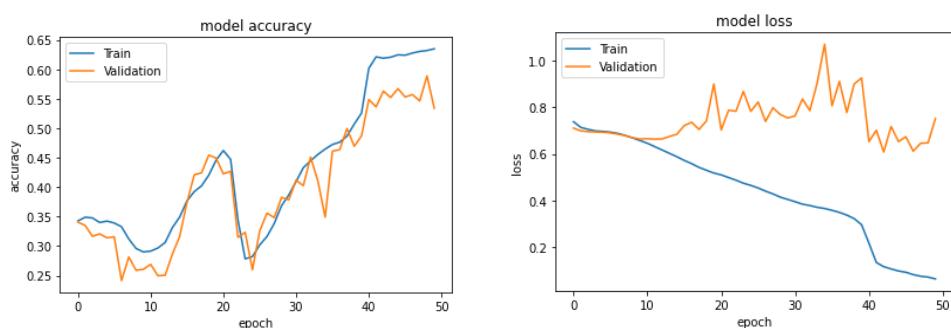
PRUEBAS PRELIMINARES DEL CLASIFICADOR ESG

Antes de probar los modelos de *HuggingFace* para el clasificador ESG se probaron un par de modelos BERT preentrenados de *TensorFlow Hub*.

Estas primeras pruebas se realizaron con un conjunto de datos estratificados del 10% de la población de la muestra total, que posteriormente se dividió en 85% entrenamiento y 15% test, ya que aquí la idea era hacer unas primeras pruebas con modelos ‘BERT’.

- Multilabel, small bert en uncased L-12 H-768 A-12

Graficas de entrenamiento



43. Gráficas de entrenamiento de BERT (1)

- Salida codificada, modelo ‘small bert en uncased L-12 H-768 A-12’:
Se ha probado a codificar la salida ‘multilabel’ como un único valor para probar qué tal serían los resultados con una sola neurona de salida de la siguiente manera:

E	S	G	Salida
Flase	Flase	Flase	0
Flase	Flase	True	1
Flase	True	Flase	2
Flase	True	True	3
True	Flase	Flase	4
True	Flase	True	5
True	True	Flase	6
True	True	True	7

Tabla 10 Valores de codificación

Gráficas de entrenamiento

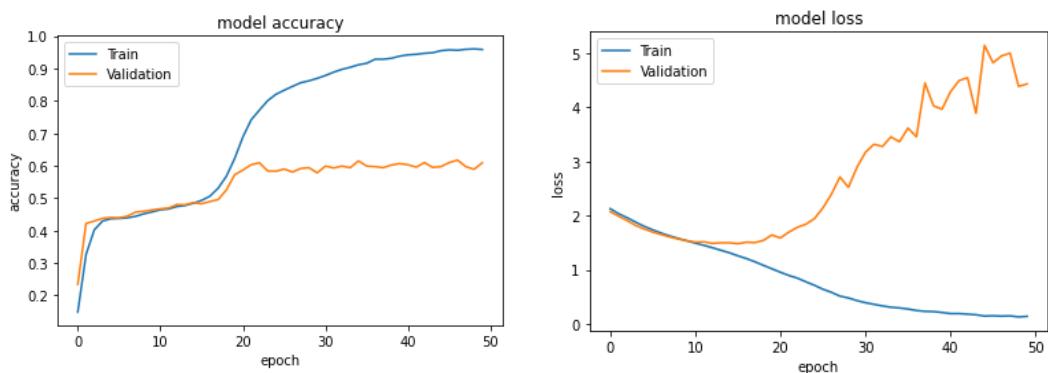


Tabla 11. Gráficas de entrenamiento de BERT (3)

En estas primeras pruebas se han entrenado con pocos datos y demasiadas iteraciones, lo que ha llevado a *overfitting* por ‘*catastrophical forgetting*’.



15. DESPLIEGUE A PRODUCCIÓN

Para la puesta en producción se ha decidido dividir el sistema en dos; el frontend y el backend. Se ha llamado frontend en este caso a la interfaz con el usuario que se encarga de la recepción de los datos del usuario y la visualización de las predicciones; El frontend realiza un procesado mínimo sobre los datos.

Hemos llamado backend a la parte que realiza el procesado de los datos proporcionados por el usuario; El backend implementa el pipeline de inferencia, es decir, procesa los datos y devuelve las predicciones al frontend.

Ambas partes del sistema se podrían ejecutar perfectamente en el mismo equipo. Sin embargo, sus requerimientos son muy diferentes; el frontend requiere de las especificaciones de un servidor web y el backend requiere de un equipo optimizado para la inferencia (GPU, TPU o similar). Es por esto por lo que se ha decidido implementar ambos sistemas de forma independiente donde la comunicación entre ambos se realiza por medio de una API.

La arquitectura para la puesta en producción se muestra en la figura siguiente.

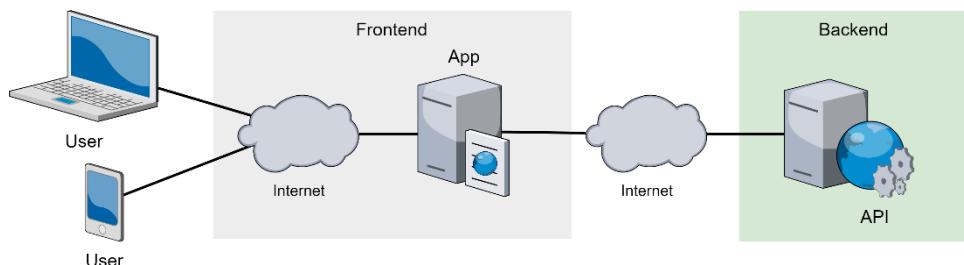


Ilustración 44 - Arquitectura del sistema en producción

Esta segmentación también simplifica el desarrollo continuo del sistema ya que se pueden actualizar de forma independiente.

En esta sección se describen la implementación de tanto el backend como el frontend así como la puesta en marcha de ambos a modo de demostración.

BACKEND API

El backend implementa el pipeline descrito en la sección 8. En concreto, dada una lista de URL de noticias, el backend devuelve, por cada noticia; su etiqueta ESG, el sentimiento y la empresa relevante.

El diagrama siguiente muestra las distintas secciones del pipeline implementado en el backend.

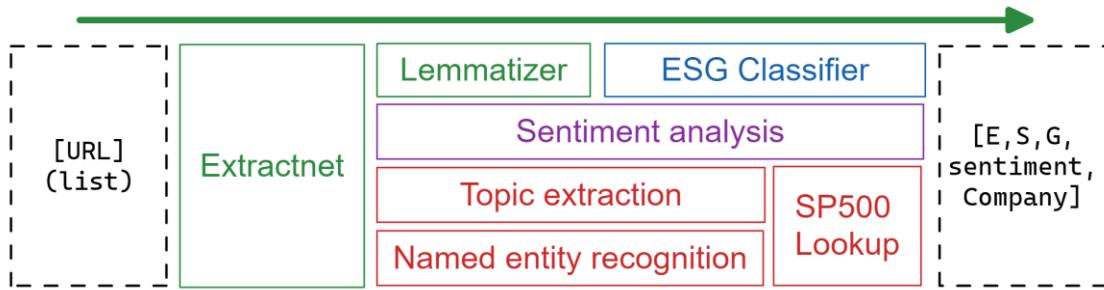


Ilustración 45 - Pipeline del backend

EXTRACCIÓN DE CONTENIDOS

La extracción de contenidos se ha implementado mediante extractnet⁴⁷.

CLASIFICADOR ESG

Los datos extraídos se han preprocesado como se ha indicado anteriormente.

La inferencia del clasificador ESG (sección 14) se ha realizado haciendo uso de la framework ONNX⁴⁸. Es decir, el modelo clasificador entrenado se ha exportado a un binario en formato ONNX y la inferencia se ha realizado utilizando la framework de ONNX.

ANÁLISIS DE SENTIMIENTO

Para el análisis de sentimiento se ha utilizado un modelo ya existente en inferencia como se ha comentado en la sección 12.

EXTRACCIÓN DE EMPRESAS

Para obtener las empresas más relevantes de la noticia se utiliza un modelo NER el cual devuelve todas las empresas detectadas y a estas empresas detectadas se les realiza dos filtrados; por su aparición en un listado del SP500 descartándose las que no formen parte de este y un filtrado en función de que tema detectado de la noticia sea relevante del sector de la empresa.

Para la extracción de empresas se ha utilizado el modelo NER (sección 13) binario. El modelo para spacy⁴⁹ se ha publicado en huggingface a modo de python wheel (whl). Este modelo se ha instalado en el backend y se utiliza en inferencia mediante la función pipe de spacy.

Para el filtrado por *topic* se ha utilizado el modelo binario joblib de BERTopic utilizando para su inferencia la librería BERTopic⁵⁰.

⁴⁷ <https://github.com/currentslab/extractnet>

⁴⁸ <https://onnx.ai/>

⁴⁹ <https://spacy.io/>

⁵⁰ <https://maartengr.github.io/BERTopic/index.html>

El filtrado por su presencia en SP500 se ha realizado mediante una simple tabla con las empresas del SP500 y sus respectivos sectores.

FRONTEND

En el frontend el usuario proporciona una serie de URL de noticias y espera obtener un análisis ESG en forma de gráficas y tablas aceptando filtrado por empresas y/o sectores. Internamente el frontend recibe la entrada del usuario mediante un fichero csv, hace una petición de predicción al backend API usando estos datos, y procesa y representa los datos recibidos del backend.

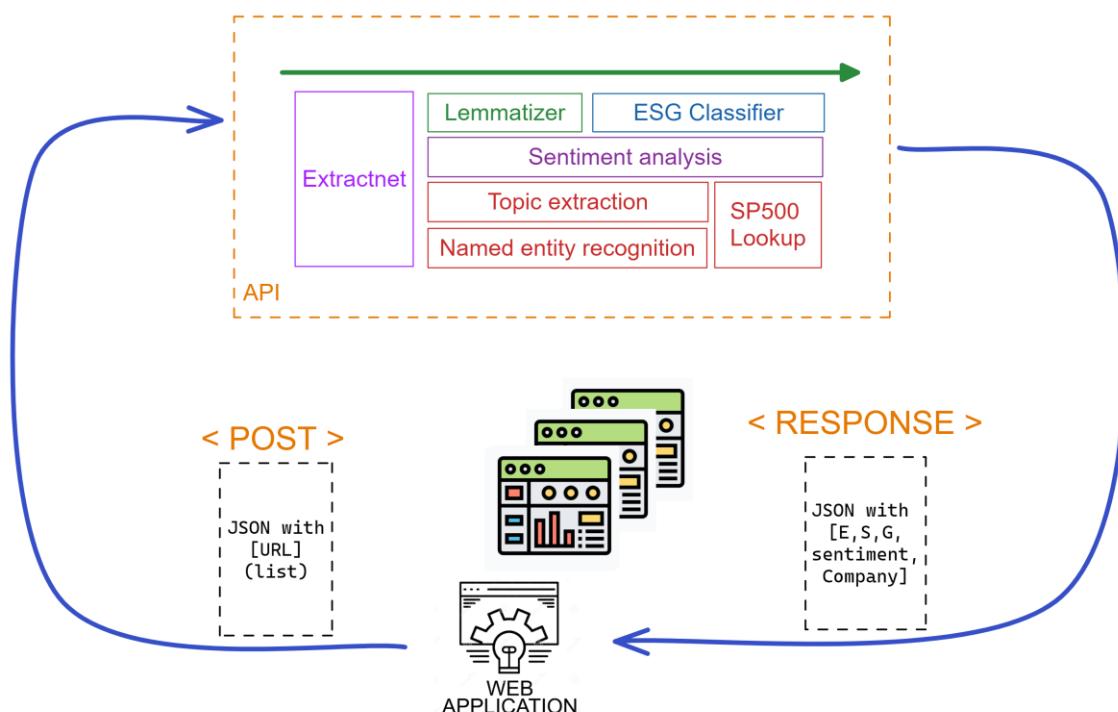


Ilustración 46 - Diagrama de flujo de conexión del frontend - backend

PUESTA EN MARCHA

En esta sección se describe como se ha puesto en marcha una demostración del sistema descrito.

FRONTEND

El frontend se ha implementado a modo de aplicación streamlit⁵¹. Esta aplicación hace uso del backend API.

⁵¹ <https://streamlit.io/>



A continuación, se muestran capturas de pantalla:



Intelligent Data Analysis Laboratory

MIA3 - Master en Inteligencia Artificial Avanzada y Aplicada

-- Data Analysis --

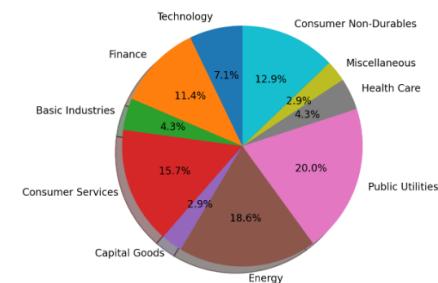
Predicted Data Filtered

	E	S	G	URL	sent_lbl	sent_score
0	0.0685	0.899	0.4565	https://www.bbc.com/news/uk-82732447	neutral	0.713
1	0.0023	0.9709	0.8514	https://www.bbc.com/news/business-62747401	negative	0.972
2	0.0027	0.9870	0.1953	https://www.bbc.com/news/technology-62744558	negative	0.928
3	0.9517	0.9852	0.0317	https://www.bbc.com/news/science-environment-42758811	negative	0.640
4	0.6422	0.2248	0.8870	https://www.theguardian.com/business/2022/sep/02/nord-st	negative	0.006
5	0.4594	0.8123	0.9355	https://www.bbc.com/news/world-europe-62706867	negative	0.913
6	0.9583	0.9108	0.9510	https://www.bbc.com/news/business-62524031	neutral	0.857
7	0.3378	0.9877	0.9505	https://www.bbc.com/news/business-62728821	negative	0.935
8	0.9520	0.9945	0.4430	https://www.bbc.com/news/science-environment-42680423	neutral	0.699
9	0.0023	0.9709	0.8514	https://www.bbc.com/news/business-62747401	negative	0.972

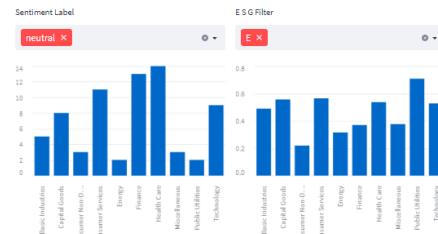
Companies Distribution



Companies Distribution



Sentiment Score Average ESG Average



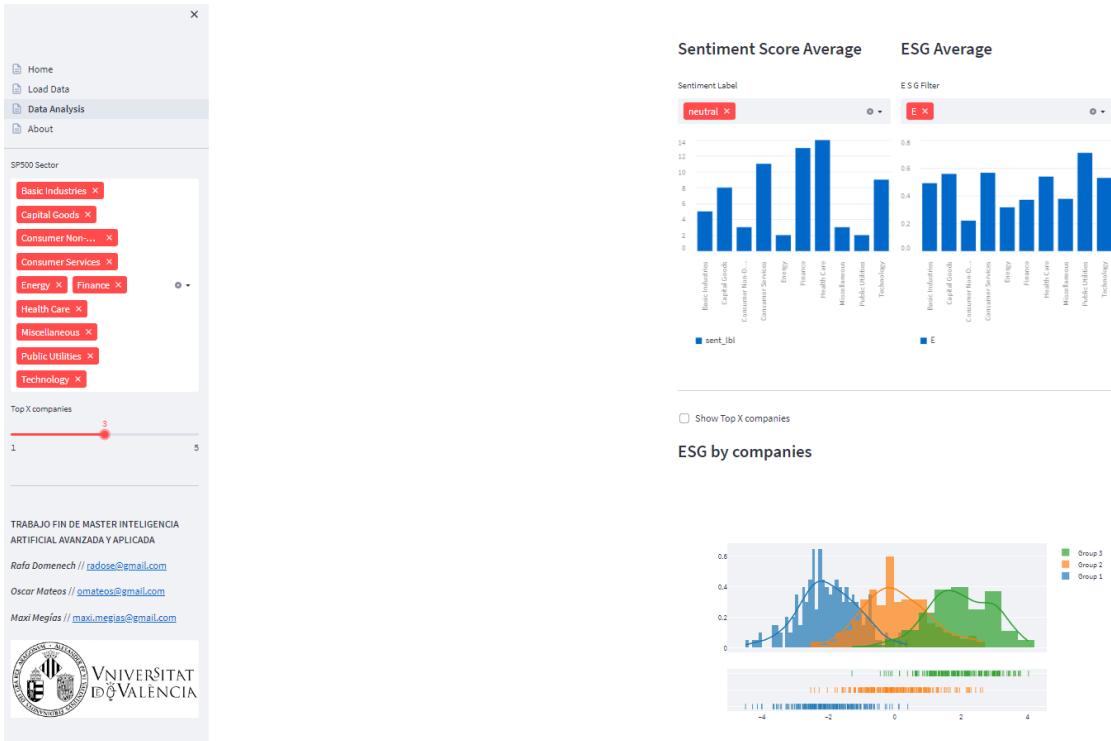


Ilustración 47. Captura pantallas webApp.

BACKEND

La API del frontend se ha programado en python usando gradio⁵² ya que, además de generar un API, proporciona una interfaz web simple muy útil durante el desarrollo. Este frontend se ha alojado en un huggingface spaces⁵³ gratuito. El backend se ha hecho público⁵⁴ y se puede utilizar haciendo un post de un JSON (formateado como se detalla a continuación) a https://hf.space/embed/ESG-TFM-UV/ESG_API_BATCH/+api/predict

```
{"data": [ {"data":[[ 'URL0' ], ... ,[ 'URLN' ]]], #2D list of URL strings "url", #Type of data, for url data use 'url' False, #When true the cached version of archive.org is used instead <integer> #Limit the companies detected }]
```

Tabla 12 - Formato JSON POST

⁵² <https://gradio.app/>

⁵³ <https://huggingface.co/spaces>

⁵⁴ https://huggingface.co/spaces/ESG-TFM-UV/ESG_API_BATCH



El backend responde con un JSON formateado a modo “dataframe” de la siguiente manera:

```
{"data": [  
    {"data":<2D list>, #2D list of strings  
     "headers":<list> #list of strings with header titles  
    },  
    ],  
  "duration": <float> # number of seconds it took to run function call  
}
```

Tabla 13 - Formato JSON response



16. CONCLUSIONES Y POSIBLES MEJORAS

Como posibles mejoras se consideran las siguientes:

- Utilizar datos mejor seleccionados, fuentes de noticias y mayor horizonte temporal para el entrenamiento del clasificador ESG, filtrado inteligente de contenidos.
- Extracción de palabras clave del texto e incluirlas en el clasificador ESG como otra fuente para que ayude a la clasificación o como entrada para el modelo de generación de tópicos.
- Uso de Transformers para modelos largos como ‘Longformer’, ‘Reformer’ o ‘BigBird’.
- Mejores tópicos para asociación semántica del contenido del texto y la selección de las empresas mediante selección de noticias específicas para cada sector y con mayor horizonte temporal.
- Pre-entrenado adicional del transformer para el clasificador: Los modelos transformer están entrenados con corpus genéricos con una distribución necesariamente diferente a la de nuestro conjunto de datos. Para mejorar esto se podría pre-entrenar el transformer en su tarea original (por ejemplo, BERT en masked language model & next sentence prediction) utilizando un conjunto de datos de distribución similar a la de interés. En este caso un preentrenamiento con un conjunto de datos de noticias, que no haya visto el modelo previamente, debería mejorar el resultado.
- Mejorar los tópicos mediante la extracción de las palabras clave de cada noticia y usarlas como entrada a BERTopic.



17. REFERENCIAS

1. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
2. Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification? CoRR abs/1905.05583 (2019). *arXiv preprint arXiv:1905.05583*.
3. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
4. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
5. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. CoRR abs/1706.03762 (2017).
6. Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: multi-label confusion matrix. *IEEE Access*, 10, 19083-19095.
7. Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.
8. Procesado del Lenguaje Natural - Dr. Francisco Martínez Martínez
9. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
10. Sarzynska-Wawer, J., Wawer, A., Pawlak, A., Szymanowska, J., Stefaniak, I., Jarkiewicz, M., & Okruszek, L. (2021). Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304, 114135.
11. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140), 1-67.
12. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, November). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In International Conference on Machine Learning (pp. 11328-11339). PMLR.
13. Lin, C. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *ACL 2004*.
14. Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).
15. Patel, Jay. (2020). Web Scraping in Python Using Beautiful Soup Library. 10.1007/978-1-4842-6576-5_2. - [Beautiful Soup: We called him Tortoise because he taught us. \(crummy.com\)](#)
16. Peters, Matthew & Lecocq, Dan. (2013). Content extraction using diverse feature sets. 89-90. 10.1145/2487788.2487828. - <http://www2013.w3c.br/companion/p89.pdf>
17. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
18. Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.



19. Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 17283-17297.
20. Nallapati, Ramesh & Zhou, Bowen & Dos Santos, Cicero & Gulcehre, Caglar & Xiang, Bing. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. 280-290. 10.18653/v1/K16-1028.
21. Grusky, M., Naaman, M., & Artzi, Y. (2018). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. arXiv preprint arXiv:1804.11283.
22. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
23. Loshchilov, I., & Hutter, F. (2017). Fixing Weight Decay Regularization in Adam. ArXiv, abs/1711.05101.
24. Shazeer, N. & Stern, M. (2018). Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. *Proceedings of the 35th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 80:4596-4604 Available from <https://proceedings.mlr.press/v80/shazeer18a.html>.
25. Dettmers, T., Lewis, M., Shleifer, S., & Zettlemoyer, L. (2021). 8-bit Optimizers via Block-wise Quantization. arXiv preprint arXiv:2110.02861.
26. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140), 1-67.
27. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, November). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning* (pp. 11328-11339). PMLR.
28. Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 3-26.
29. Dixit, K., & Al-Onaizan, Y. (2019). Span-Level Model for Relation Extraction. *ACL*.
30. Eberts, M., & Ulges, A. (2019). Span-based joint entity and relation extraction with transformer pre-training. arXiv preprint arXiv:1909.07755.
31. Kalyanathaya, Krishna & D., Akila & G., Suseendran. (2019). A Fuzzy Approach to Approximate String Matching for Text Retrieval in NLP. *Journal of Computational Information Systems*. 15. 26-32.
32. Haldar, R., & Mukhopadhyay, D. (2011). Levenshtein distance technique in dictionary lookup methods: An improved approach. arXiv preprint arXiv:1101.1232.



33. LISTADO DE ILUSTRACIONES Y TABLAS

ILUSTRACIONES

Ilustración 1. Criterios ESG [Mukut Mukherjee - https://towardsdatascience.com/nlp-meets-sustainable-investing-d0542b3c264b]	12
Ilustración 2. Estado del arte de los modelos Transformer NLP 20/21	16
Ilustración 3. Extracto dataset original.	19
Ilustración 4. Flujo del modelo TFM CRITERIOS ESG.....	21
Ilustración 5. Ejemplo de campos del dataset de extracción inicial	23
Ilustración 6. Distribución de noticias por día	25
Ilustración 7. Distribución de clases.....	25
Ilustración 8. Estudio X:num_words Y:news (Batch 10%).....	26
Ilustración 9. Estudio frecuencia de palabras.	26
Ilustración 10. BoxPlot de longitud del texto vs ESG_Encoded.	27
Ilustración 11.Distribución de longitudes de palabras (1)	28
Ilustración 12.Distribución de longitudes de palabras (2)	28
Ilustración 13. Distribución por longitud de palabras.....	29
Ilustración 14. Estudio frecuencial de palabras	30
Ilustración 15. Comparativas modelos Seq2Seq LSTM & LSTM Bidirectional	34
Ilustración 16. Comparativa Rouge Score de modelos Seq2seq.....	35
Ilustración 17. Modelo Seq2seq LSTM	36
Ilustración 18. Modelo Seq2seq LSTM Bidireccional	36
Ilustración 19.Tareas del Modelo T5.....	38
Ilustración 20. Arquitectura Pegasus	38
Ilustración 21. Entrenamiento T5	41
Ilustración 22. Entrenamiento Pegasus (1)	41
Ilustración 23. Entrenamiento Pegasus (2)	43
Ilustración 24. Ejemplo de anotación con doccano	47
Ilustración 25. Error del transformer	49
Ilustración 26. Error de tok2vec.....	49
Ilustración 27. Error del spancat	50
Ilustración 28. F1-score de los tres modelos	50
Ilustración 29. <i>Precision</i> de los tres modelos.....	51
Ilustración 30. <i>Recall</i> de los tres modelos.....	51



Ilustración 31. Puntuación promedio de los tres modelos	52
Ilustración 32. Ejemplo de salida de modelo de span-categorizer	53
Ilustración 33. Ejemplo de entrenamiento de BERTTopic	55
Ilustración 34. Visualización de tópicos (1).....	56
Ilustración 35. Visualización de tópicos (2).....	57
Ilustración 36. Ejemplo de asociación semántica (1).....	58
Ilustración 37. Ejemplo de asociación semántica (2).....	59
Ilustración 38 - Diagrama del modelo clasificador ESG	60
Ilustración 39 - Sugerencia de la utilidad "learning rate finder".....	62
Ilustración 40 - Estudio de tasa de aprendizaje	63
Ilustración 41 - Diagrama de la configuración final de la red neuronal densa	64
Ilustración 42 - Matriz de confusión y su versión normalizada obtenida con el mejor modelo clasificador sobre el conjunto de test	68
43. Gráficas de entrenamiento de BERT (1).....	68
Ilustración 44 - Arquitectura del sistema en producción.....	70
Ilustración 45 - Pipeline del backend	71
Ilustración 46 - Diagrama de flujo de conexión del frontend - backend	72
Ilustración 47. Captura pantallas webApp.....	74

TABLAS

Tabla 1. Detalle de las columnas del dataset original.....	19
Tabla 2. ESG Encoded.....	27
Tabla 3. Captura VS Code	32
Tabla 4. Texrank Rouge score según el artículo referenciado con dataset DUC.....	32
Tabla 5 - Comparativa de distintos modelos transformer	65
Tabla 6 - Comparativa entre texto lematizado y sin lematizar	65
Tabla 7 - Comparativa de distintos métodos de truncado.....	66
Tabla 8 - Comparative de distintos métodos de generación de resumen	67
Tabla 9 - Mejores resultados del modelo clasificador ESG	67
Tabla 10 Valores de codificación.....	69
Tabla 11. Gráficas de entrenamiento de BERT (3)	69
Tabla 12 - Formato JSON POST	74



Tabla 13 - Formato JSON response 75

