STOP FAKE NEWS

Natalia Sánchez y Pablo Sánchez

Fases del proyecto:

- Recogida de datos
- Análisis y preprocesamiento de los datos
- Entrenar y evaluar modelos
- Comparar y elegir los modelos con mejores resultados
- Crear una interfaz que permita interactuar con el modelo

INTRODUCCIÓN

Stop Fake News es el proyecto de data science hecho y presentado por Pablo Sánchez Ochando y Natalia Sánchez Carretero.

La idea de este proyecto surge al reparar en la importancia de tener acceso a información veraz y contrastada en un mundo que nos bombardea diariamente con una cantidad inabarcable de contenido. Discriminar lo verdadero de lo falso es más difícil que nunca debido a veces a la manipulación deliberada de los medios, y en otras ocasiones a la falta de herramientas para comprobar las fuentes de los datos.

En definitiva, este proyecto no se dirige a un colectivo en concreto ya que su aplicación es útil para cualquiera que necesite comprobar la veracidad de los hechos que se publican.

Los objetivos principales del proyecto son: **detectar noticias falsas**, lo que derivó en poder **predecir el periódico que publica la noticia** y por último, **crear una interfaz** capaz de ejecutar estos modelos para que cualquier usuario pueda utilizarlo.

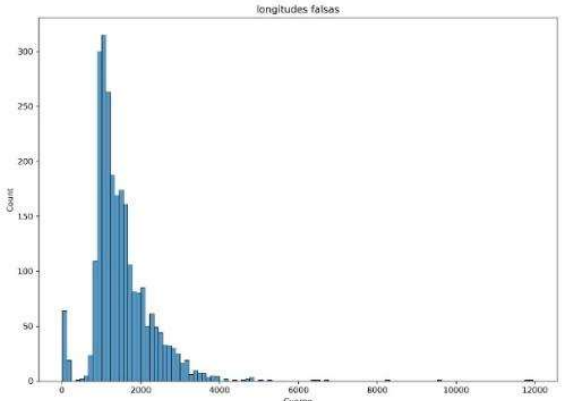
1. RECOGIDA DE DATOS

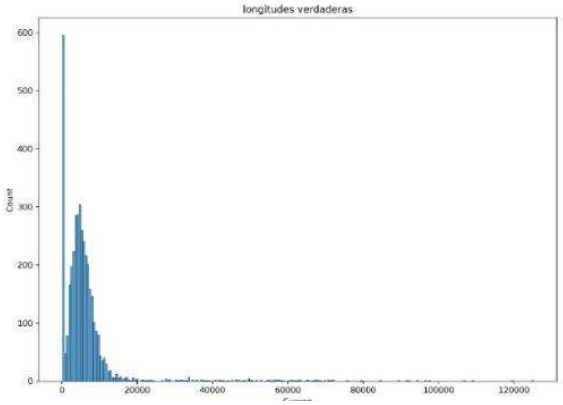
Para recoger un número considerable de noticias falsas, en este apartado lo ideal hubiera sido recopilar noticias que fuesen realmente falsas, aunque suene paradójico, es decir, noticias que al ser publicadas se consideraron verdaderas hasta que alguien las desmintió. Sin embargo, debido a la enorme complejidad de reunir la cantidad necesaria de noticias de este tipo, ya que no están agrupadas en ningún sitio, hemos optado por recoger estas noticias de dos periódicos conocidos por publicar únicamente noticias falsas en tono humorístico: **ElMundoToday** y **HayNoticia.es**. Para las noticias verdaderas no hubo este problema y elegimos tres periódicos intentando que tuvieran diferentes tendencias políticas: **El País, El Mundo y Eldiario.es.**

De estos 5 periódicos hacemos el web scraping, que es la técnica que usamos para la extracción de los datos de una página web. Para ello se han utilizado librerías como **beautifulsoup**, **requests**, y **regex** de expresiones regulares. De cada noticia se ha extraído el **periódico**, el **hipervínculo**, el **titular**, **subtítulo**, **cuerpo, fecha de publicación** y **categoría** de la noticia. Todos estos datos eran importantes para que hubiera la mayor variedad y objetividad posible.

2. ANÁLISIS Y PREPROCESAMIENTO DE LOS DATOS

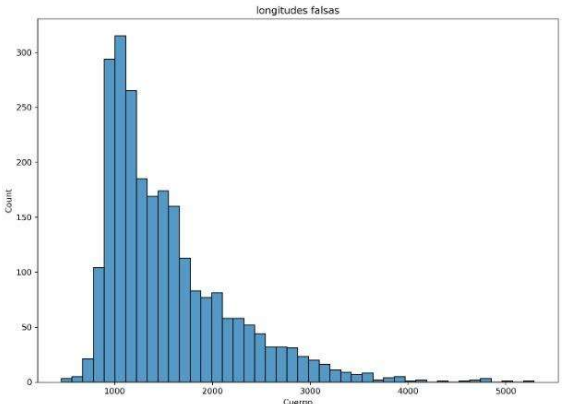
Tras una primera limpieza rápida, observamos gran disparidad en las longitudes de los cuerpos de los dos tipos de noticias: las verdaderas eran más de cuatro veces más largas que las falsas de media. Véase, como dato, que las noticias cuyo cuerpo superaba los 4000 caracteres representaban algo más del 60% en las verdaderas, mientras que ni llegaban al 1% en las falsas. Asimismo, la distribución de las longitudes en las noticias verdaderas era fuertemente asimétrica, y con un coeficiente de variación tres veces superior al de las falsas.

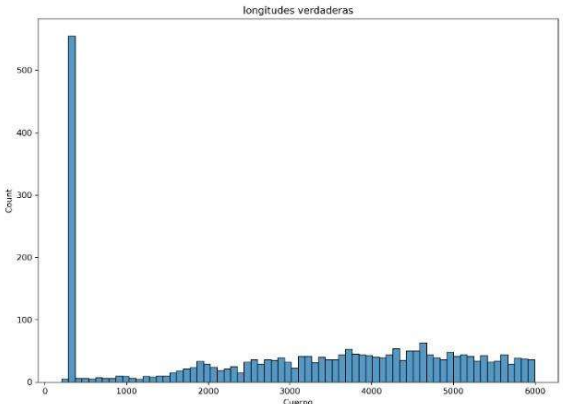




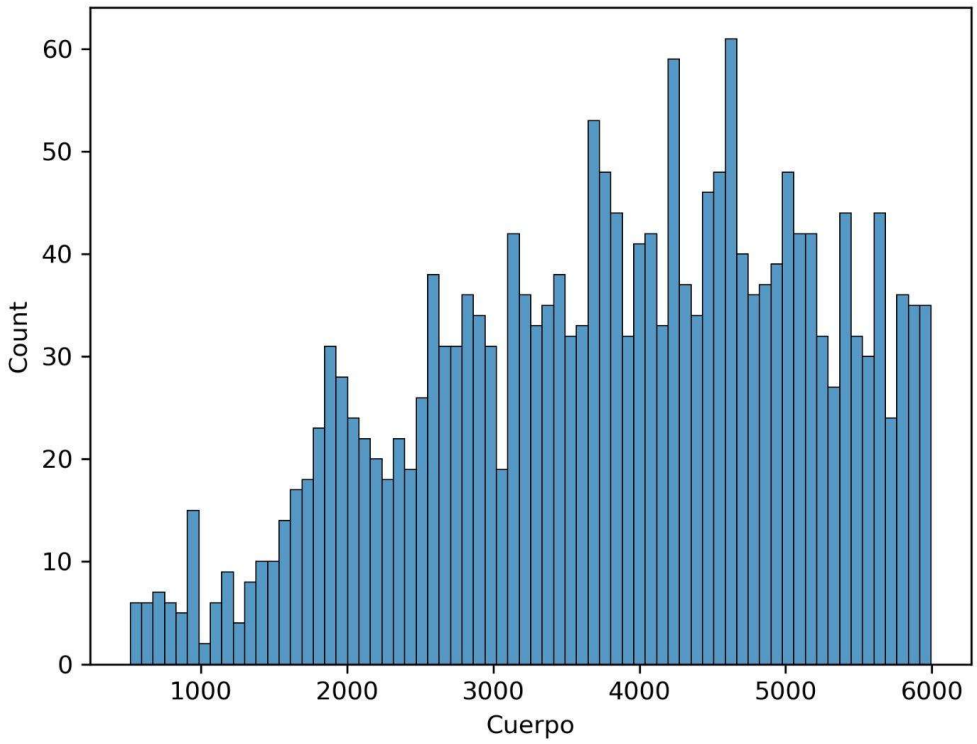
Esto solo podía traer sesgo, sobreajuste y demás consecuencias negativas al modelo, por lo que decidimos eliminar todas las noticias con longitudes por encima de los 6000 caracteres y por debajo de 200 (había “noticias” con cuerpo vacío o con longitudes

mínimas que ni siquiera tenían estructura de noticia como tal). Los histogramas quedaban tal que así:

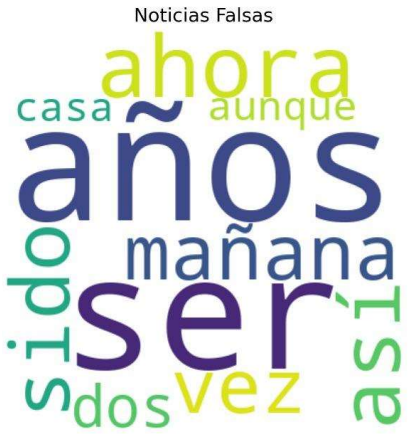




Se puso de relieve que existía un grupo grande de noticias con longitudes cortas y muy similares. Era probable que este patrón no fuera el único que existiera en un conjunto tan homogéneo y, en efecto, hallamos otro: todas estaban cortadas, desde la mitad aproximadamente, por la fórmula “Hazte Premium…”. Eran noticias incompletas a las que solo se tenía completo acceso mediante una cuenta premium. Descartamos estas 600, volvimos a hacer web scraping y volvimos a descartar algunas hasta que conseguimos un balance entre cantidades de un tipo y otro de noticias, y una distribución de longitudes con mejores propiedades para las verdaderas.



Por último, llevamos a cabo un análisis de palabras más frecuentes, utilizando la clase *CountVectorizer* de la librería *scikit-learn* de Python. Esto nos ayudó a ver que, aunque en tema de longitudes sí había bastante disparidad, no parecía que el vocabulario fuera a ser un diferenciador claro entre ambos tipos de noticias. También nos ayudó a considerar palabras a incluir en las listas de "stopwords" utilizadas más adelante.



3. ENTRENAR Y EVALUAR MODELOS

Como primera iteración del modelo predictivo que buscábamos, utilizamos el vectorizador TF-IDF junto con una Regresión Logística, tanto para predecir la veracidad de las noticias como el periódico al que pertenecían. Probamos también algunos otros modelos, como árboles de decisión, Naive Bayes, Ada Boost o Gradient Boosting. Los resultados ya eran mejor de lo que esperábamos, aun con esta combinación de vectorizador y modelo que no capturaba suficientemente la complejidad de los datos. Como representante de esta fase, escogimos la Regresión Logística, pues proporcionaba muy buenos resultados con gran rapidez. El punto negativo es que, como el resto de modelos antes mencionados, quedaba algo sobreajustado incluso corrigiendo hiperparámetros. La solución más plausible a este punto la proponemos en el último apartado de este documento.

También hemos utilizado el modelo Doc2Vec, también conocido como Paragraph Vector que es una técnica de aprendizaje automático desarrollada para representar documentos de texto en forma de vectores densos y continuos. Es una extensión de Word2Vec, que se utiliza para representar palabras como vectores distribuidos y capturar relaciones semánticas entre ellas.

La idea detrás de Doc2Vec es obtener representaciones vectoriales para documentos completos, como oraciones, párrafos o documentos completos, de manera similar a cómo Word2Vec genera representaciones vectoriales para palabras.

Una vez entrenado, el modelo Doc2Vec genera un vector de características único para cada documento, que representa su contenido semántico.

Después elegimos un modelo de aprendizaje automático adecuado para la tarea de clasificación binaria (verdadero o falso). En este caso, hemos usado la regresión logística ya que habíamos comprobado que daba los mejores resultados.

3.1. Transformers

Ya con un modelo base que, aunque se podría mejorar, proporcionaba predicciones con cierta solidez y era fácil de manejar en todos los sentidos, pasamos a investigar cuáles de los grandes modelos de lenguaje (LLM por sus siglas en inglés) encajarían con nuestras necesidades. Una primera selección incluía BERT, DistilBERT, RoBERTa y XLNET, entre otros. Finalmente sólo fuimos capaces de entrenar y utilizar satisfactoriamente DistilBERT, BERT-base y Albert (en una versión preentrenada en español). Incluso con éstos tuvimos dificultades para hacer *fine tuning* con nuestros datos, debido a la gran demanda de recursos computacionales que presentan estas redes profundas. Finalmente bastó con algunos ajustes en el *batch size* del entrenamiento para conseguir que funcionaran.
Los mejores resultados los conseguimos con el modelo Albert, a pesar de ser el más pequeño de los tres. Todos ellos resultaron más precisos que el modelo base (TF-IDF + Regresión Logística) y con menor sobreajuste, como cabría esperar.

4. INTERFAZ PARA INTERACTUAR CON EL MODELO

Con el propósito de que el usuario pueda valerse de nuestro trabajo de una forma sencilla y sin posibilidad de alterar el código, conectamos el modelo Albert, entrenado con todos nuestros datos, a una interfaz gráfica creada con la librería Streamlit de Python.

5. CONCLUSIONES

Consideramos que los resultados son buenos, mejores incluso de lo esperado. Sin embargo, es evidente que hay muchas mejoras que realizar y muchas direcciones en las que hacerlo. Por citar algunas de ellas:

- Mayor recopilación de datos. Un modelo entrenado en una cantidad mayor y más variada de datos tendría el poder de generalizar mejor y ser preciso en una mayor variedad de periódicos (idealmente en todos).

- Mayor profundidad en la limpieza y análisis de los datos. No es solo la cantidad sino la calidad.

- Explorar otros modelos de Deep Learning para clasificación de texto y/o optimizar el rendimiento de los considerados hasta el momento.

- Conseguir noticias realmente falsas, si fuera posible.

- Crear una versión del modelo en inglés u otros idiomas.

- Crear una versión entrenada en un conjunto de noticias políticas de varios periódicos, etiquetadas según ideología. Sería interesante ver qué proporción de noticias de cada ideología predice el modelo para cada periódico.