UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Clasificación de textos basado en los modelos pre-entrenados BERT

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Manuel Sánchez Mascarell

Tutor: Francisco Casacuberta Nolla

Curso 2020-2021

Resumen

El objetivo principal de este proyecto de fin de grado, consiste en el estudio de las técnicas de procesamiento del lenguaje natural.

En concreto, se revisa una red neuronal, entrenada por Google y denominada Bert.

Bert, ha sido entrenado con un conjunto de datos enorme, y con ello, es capaz de procesar textos y obtener información sobre cada una de las palabaras que lo forman y el contexto en el que se encuentran.

Partiendo de Bert, se diseñan e implementan varios modelos con aplicaciones en el mundo real, que permiten automatizar tareas relacionadas con el procesamiento de textos.

Para la realización de los modelos, se emplean técnicas de inteligencia artificial y machine learning.

Palabras clave: Procesamiento del lenguaje natural, red neuronal, procesamiento de texto, inteligencia artificial, machine learning

Resum

L'objectiu principal d'aquest projecte de fi de grau, consisteix en l'estudi de les tècniques de processament de el llenguatge natural.

En concret, es revisa una xarxa neuronal, entrenada per Google, anomenada Bert.

Bert, ha estat entrenat amb un conjunt de dades enorme, i amb això, és capaç de processar textos i obtenir informació sobre cadascuna de les paraules que el formen i el context en què es troben.

Partint de Bert, es dissenyen i implementen diversos models amb aplicacions en el món real, que permeten automatitzar tasques relacionades amb el processament de textos.

Per a la realització dels models, s'empren tècniques d'intel·ligència artificial i machine learning.

Paraules clau: Processament del llenguatje natural, xarxa neuronal, processament de text, intel·ligència artificial, machinelearning

Abstract

The main objective of this end grade project consists in the study of the techniques of natural languaje processing.

Mainly, it is reviewed a neural network, trained by Google named Bert.

Bert, has been trained with a big ammount of data, with this, is able to process a text and obtain information about each of the words that make it up and the context they are.

Starting from Bert, some models with actual world aplication, are designed, those allows automate tasks related with text processing.

For make the models, techniques of inteligence artificial and machine learning are used.

Key words: Natural languaje processing, neural network, text processing, artificial inteligence, machine learning

Índice general

Índice general v
Índice de figuras vii

1 Introducción 1
1.1 Prefacio . . . 1
1.2 Contexto y justificación del trabajo . . . 1
1.3 Objetivos del proyecto . . . 2
1.4 Expectativas . . . 2
1.5 Lenguaje de programación y recursos . . . 3
1.6 Estructura de la memoria . . . 3
1.7 Conceptos básicos . . . 3

2 ¿Qué es BERT? 5
2.1 Historia . . . 5
2.2 Bidireccionalidad . . . 6
2.3 Redes transformer . . . 7

3 Clasificación de Valoraciones 9
3.1 Estado del arte para el Análisis de Valoraciones . . . 9
3.2 Motivación . . . 9
3.3 Evaluación del Modelo . . . 10
3.3.1 Google App Reviews . . . 10
3.3.2 App Reviews . . . 11
3.4 Clasificador de Valoraciones . . . 13

4 Clasificación de Noticias 17
4.1 Estado del arte para la clasificación de Noticias . . . 17
4.2 Motivación . . . 17
4.3 Evaluación del modelo . . . 18
4.3.1 Conjunto de noticias . . . 18

5 Marco experimental 21
5.1 Tokenizador de Bert . . . 21
5.2 Experimento a partir de los resultados de Bert . . . 21
5.2.1 Introducción . . . 21
5.2.2 Aplicación en el proyecto . . . 22
5.2.3 Estructura del experimento . . . 22
5.2.4 Resultados para el clasificador de valoraciones . . . 24
5.2.5 Resultados para el clasificador de noticias . . . 25
5.3 Experimento K-BERT . . . 26
5.3.1 Introducción . . . 26
5.3.2 Aplicación en el proyecto . . . 26
5.3.3 Desarrollo de Base de conocimiento . . . 26
5.3.4 Resultados obtenidos . . . 26
5.4 Conclusiones . . . 27

6 Desarrollo del demostrador 29

6.1 Introducción 29
6.2 Clasificadores 29
6.3 Optimizaciones 32
7 Conclusiones **33**
7.1 Conclusiones del proyecto 33
7.2 Trabajos Futuros 33
7.3 Relación con los estudios cursados 34
Bibliografía **35**

Apéndices
A Modelo base para el desarrollo de los clasificadores **37**
B Procesos internos en las redes transformer **39**
C Modelo de selección por Patience **45**

Índice de figuras

1.1 NLP [1] . . . 1

2.1 Consulta buscador.[2] . . . 5
2.2 Ejemplo de bidireccionalidad. [3] . . . 6
2.3 Estructura Bert . . . 6
2.4 Estructura base de Bert . . . 7

3.1 Distribución del conjunto Google App Reviews . . . 10
3.2 Conjunto Google App Reviews redistribuido . . . 11
3.3 Resultados obtenidos Google App Reviews . . . 11
3.4 Distribución del conjunto App Reviews . . . 12
3.5 Conjunto App Reviews redistribuido . . . 12
3.6 Resultados obtenidos App Reviews . . . 13
3.7 Distribución del conjunto TripAdvisor . . . 13
3.8 Longitud valoraciones tokenizadas . . . 14
3.9 Resultados validación TripAdvisor . . . 14

4.1 Distribución del conjunto de noticias . . . 18
4.2 Longitud cabeceras tokenizadas 41 categorías . . . 18
4.3 Longitud cabeceras tokenizadas 11 categorías . . . 19
4.4 Comparativa resultados distribuciones 11 categorías y 41 categorías . . . 19

5.1 Estructura del experimento sin el token [CLS] . . . 22
5.2 Estructura del experimento con el token [CLS] . . . 23
5.3 Tratamiento de los tokens [PAD] . . . 23
5.4 Resultados experimento análisis de valoraciones sin [CLS] . . . 24
5.5 Resultados experimento análisis de valoraciones con [CLS] . . . 24
5.6 Resultados experimento clasificador de noticias sin [CLS] . . . 25
5.7 Resultados experimento clasificación de noticias con [CLS] . . . 25

6.1 Selección de modelo . . . 29
6.2 Pantalla de inicio . . . 30
6.3 Valoración de TripAdvisor . . . 30
6.4 Ejemplo de uso del clasificador de valoraciones . . . 30
6.5 Ejemplo de uso del clasificador de noticias . . . 31
6.6 Ejemplo de uso del predictor de máscaras . . . 31

B.1 Estructura CodificadorDecodificador [11] . . . 39
B.2 Ejemplo de embedding . . . 40
B.3 Embedding posicional . . . 40
B.4 Estructura del modelo . . . 41
B.5 Bloque residual . . . 42

CAPÍTULO 1
Introducción

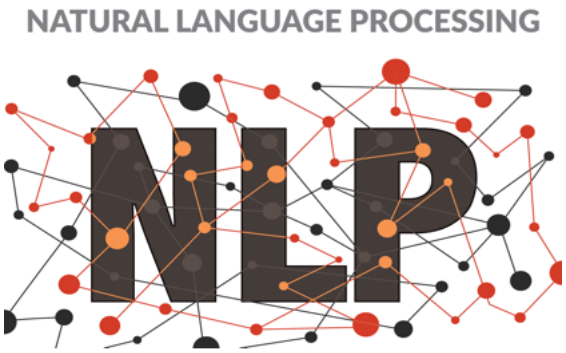


Figura 1.1: NLP [1]

1.1 Prefacio

Con el incremento de información que podemos encontrar en internet, han ido surgiendo investigaciones para desarrollar técnicas que permitan identificar y organizar este tipo de informaciones, algunas de estas técnicas, son las denominadas técnicas de procesamiento del lenguaje.

Algunas empresas muy importantes en el sector tecnológico como el caso de Google, han invertido en investigar y generar modelos capaces de procesar grandes cantidades de información, logrando el desarrollo de una red neuronal denominada Bert capaz de procesar textos.

1.2 Contexto y justificación del trabajo

En la actualidad, podemos encontrar muchas aplicaciones que parten del procesamiento del lenguaje natural también conocidos como sistemas NLP, han ido ganando mucha importancia en aplicaciones orientadas a la interacción con el usuario, como es el caso de los buscadores web o aplicaciones de asistencia técnica como los chat bots.

El rápido avance de internet, también ha favorecido en el desarrollo de estos sistemas, muchas empresas, se han interesado en obtener información sobre los productos y/o servicios que oferta, incluso son los propios usuarios los que cada vez están más interesados

en tener información sobre las experiencias ya vividas por otras personas que ya han hecho uso del producto o del servicio ofertado.

Al incrementarse la cantidad de información, manejarla por un equipo de personas, se vuelve más complicado, aquí es donde los sistemas NLP, ganan más fuerza, permitiendo agilizar este tipo de procesos y permitiendo extraer la información más relevante.

Algunas empresas como Google, OpenAI, Microsoft, Nvidia o Facebook, tienen sus desarrollos propios para este tipo de redes, y tienen equipos de investigación detrás para mejorar los modelos, reduciendo el coste de recursos del ordenador.

En el proyecto, se realiza un estudio sobre la red ya entrenada por Google denominada Bert.

Bert, es una red neuronal, que ha sido entrenada por Google utilizando un corpus de datos muy grande, formado por BookCorpus[1] y Wikipedia en Inglés[2].

Gracias a este entrenamiento, se consigue que Bert, sea capaz de reconocer que palabras aportan información en el contexto.

1.3 Objetivos del proyecto

El proyecto tiene como objetivos los siguientes,

- Estudio de la red desarrollada por Google: Bert
- Implementación de un clasificador de Noticias a partir de Bert
- Implementación de un clasificador de Valoraciones a partir de Bert
- Desarrollo de experimentos con Bert para mejorar los resultados de clasificación
- Desarrollo de una aplicación de demostración

1.4 Expectativas

Este proyecto tiene como intención, aprender más acerca de las redes neuronales y el aprendizaje automático, además de entender cómo funciona el NLP.

Así mismo, aplicar las técnicas aprendidas en el grado de informática y los conocimientos obtenidos en la rama de computación, para generar un modelo de clasificación de valoraciones y un modelo de clasificación de noticias.

Finalmente, obtener conocimientos sobre alguna de las librerías más relevantes en pyhton para la creación de modelos de inteligencia artificial

[1] https://paperswithcode.com/dataset/bookcorpus
[2] https://en.wikipedia.org/wiki/Main_Page

1.5 Lenguaje de programación y recursos

Como lenguaje de programación para la implementación del proyecto, se ha utilizado Python[3], un lenguaje interpretado de alto nivel, para la realización del proyecto, se han utilizado varias librerías adicionales,

- Pandas[4]: Manejo de los datos.
- Numpy[5]: Operaciones matriciales.
- MatplotLib[6]: Creación de gráficas.
- PyTorch[7]: Redes neuronales y funciones.
- Transformers (HuggingFace)[8]: Modelos pre-entrenados de Bert.
- Multiprocessing[9]: Optimizaciones.
- PyQt5[10]: Diseño de la interfaz gráfica.
- Sklearn[11]: Creación de los conjuntos de entrenamiento, validación y testing.

1.6 Estructura de la memoria

En esta memoria, se revisa el funcionamiento de la red neuronal peviamente entrenada por Google denominada Bert.

La memoria, está dividida en capítulos, en los iniciales, se explican las técnicas de NLP además de la tecnología que utiliza Bert, y los procesos que realiza.

Tras conocer Bert y los NLP, se presentan varias tareas que se han implementado a partir del modelo Bert, estas tareas, consisten en la clasificación de valoraciones y la clasificación de noticias.

Posteriormente, se comentan varios experimentos que se han realizado sobre estas tareas, para observar como se comportan los resultados.

Finalmente, podemos encontrar el desarrollo de una aplicación que contiene los modelos ya entrenados de las tareas básicas que se han implementado dentro del proyecto.

1.7 Conceptos básicos

A lo largo del proyecto, aparecen términos técnicos relacionados con la inteligencia artificial y los NLP, por ello para entender la memoria con mayor claridad se presentan algunos de los términos.

[3] https://www.python.org/
[4] https://pandas.pydata.org/
[5] https://numpy.org
[6] https://matplotlib.org
[7] https://pytorch.com
[8] https://huggingface.co/transformers/
[9] https://docs.python.org/3/library/multiprocessing.html
[10] https://pypi.org/project/PyQt5/
[11] https://scikit-learn.org

Tokenización: La tokenización de texto, es uno de los pasos previos a tener en cuenta antes de crear cualquier modelo de procesamiento del lenguaje.

Este proceso, suele variar la forma en la que se tratan los símbolos especiales o los espacios, normalmente, estos se eliminan, dividiendo el texto en unidades morfológicas denominadas tokens [4].

En los modelos que se van a utilizar en el proyecto, la tokenización se realiza utilizando un diccionario de términos, generar este tipo de diccionarios puede resultar muy costoso, por lo que se utiliza un algoritmo denominado WordPiece.

Epoch: Es un hiperparámetro, es decir, un parámetro que se utiliza para controlar el proceso de aprendizaje, define el número de veces que el modelo va a ser entrenado con el conjunto de datos reservado para entrenamiento [5].

Batch: Este término al igual que el término Epoch, es un hiperparámetro, en este caso, este parámetro se utiliza para indicar el número de predicciones que debe realizar el modelo para cada etapa antes de que sus parámetros se vean modificados [5].

Patience: Para la obtención de los resultados, se ha implementado un modelo de selección de la mejor etapa de entrenamiento por paciencia, la paciencia viene definida por el usuario y especifica el número de etapas que deben transcurrir tras haber obtenido el mejor resultado de validación.

NLP: Cuyas siglas provienen de *Natural Language Processing*, es un campo comprendido en las ciencias de la computación, de la Inteligencia Artificial y de la lingüística que estudia las interacciones entre los ordenadores y el lenguaje humano, se encarga de la formulación e investigación de nuevos mecanismos para la comunicación entre personas y ordenadores [4].

CAPÍTULO 2
¿Qué es BERT?

2.1 Historia

Bert, cuyas siglas significan Bidirectional Encoder for Transformers, es una red neuronal multicapa publicado por Jacob Devling et al. en 2008., tiene su origen en los modelos previamente entrenados para la representación del contexto.

El objetivo de Bert, consiste en interpretar nuestro lenguaje de una manera mucho más natural, mediante PNL (Programación Neuro Lingüística).

Hasta ahora, de acuerdo con medios como The Verge[1] y Wired[2], las búsquedas en Google tras la aparición de Bert, han mejorado en aproximadamente un 10 %, según resultados presentados por los usuarios.

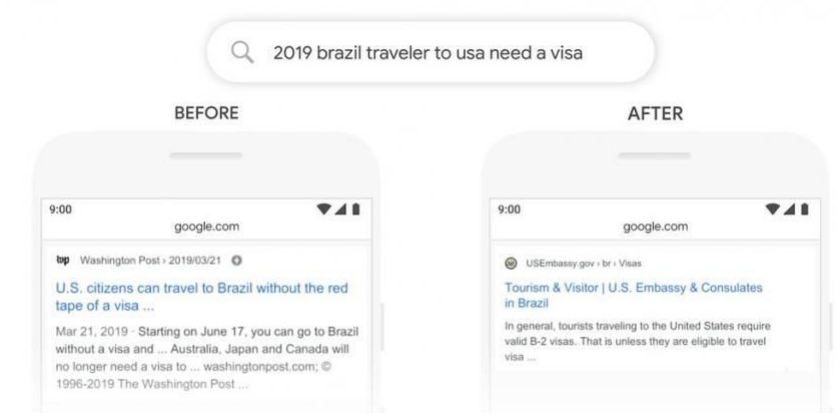


Figura 2.1: Consulta buscador.[2]

Como podemos observar en la figura 2.1, previamente al lanzamiento de Bert el algoritmo de búsqueda, no era capaz de identificar que para la consulta "2019 Brazil traveler to USA need a visa", el pasajero, busca viajes de Brasil a USA, el algoritmo, extrae la información más relevante, sin relacionarla con el contexto.

Mientras que con Bert, el nuevo buscador es capaz de saber que el pasajero está viajando de Brasil a USA, adecuando los resultados a la consulta del usuario.

[1]The verge: https://www.theverge.com/
[2]Wired: https://www.wired.com

Como predecesor a Bert en la tarea de resolver consultas, podemos destacar el algoritmo RankBrain, primer algoritmo utilizado por Google que emplea técnicas de inteligencia artificial.

2.2 Bidireccionalidad

Bert utiliza un modelo de análisis bidireccional, esto significa, que se revisan las palabras que se encuentran tanto a la derecha como a la izquierda de la palabra clave que se está analizando.

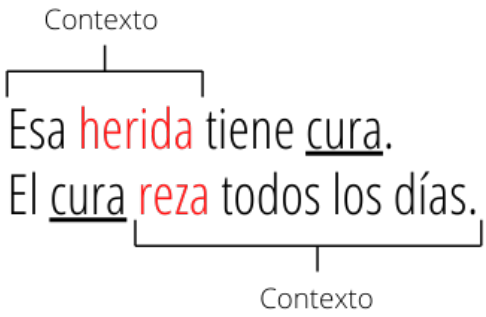


Figura 2.2: Ejemplo de bidireccionalidad. [3]

En la figura 2.2, se puede observar un ejemplo de bidireccionalidad, donde las palabras clave aparecen subrayadas, y las palabras que aportan significado a las palabras clave aparecen resaltadas en color rojo.

Como se puede observar, la palabra a analizar es una palabra polisémica.

Analizando la primera frase, se puede observar como el significado de "cura", viene dado por una palabra que le precede, mientras que, analizando la segunda frase, el significado de "cura", viene dado por una palabra que tiene a continuación.

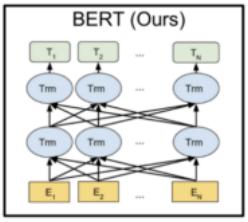


Figura 2.3: Estructura Bert

El modelo representado en la figura 2.3 nos permite apreciar como Bert, sigue una estructura multicapa donde como entrada en cada una de las células, se utilizan todas las componentes de la etapa anterior, tomando, en la primera capa, todas las palabras que forman el texto.

Esta conexión puede observarse en la figura a través de las aristas que representan el flujo de información.

2.3 Redes transformer

La arquitectura que utiliza Bert es la denominada arquitectura transformer, esta fué presentada por Google a finales del 2017, en una presentación conocida como "Attention is all you need", la aparición de esta arquitectura, trajo consigo las denominadas como capas de auto-atención.

Las capas de auto-atención, se encargan de codificar cada palabra en función de las demás que forman el texto, permitiendo, de esta manera, se consigue abstraer la información del contexto y representarlo de forma numérica.

Las redes transformer, siguen una estructura codificador decodificador.

Los procesos internos que realizan las redes transformer se encuentran detallados en el Apéndice B.

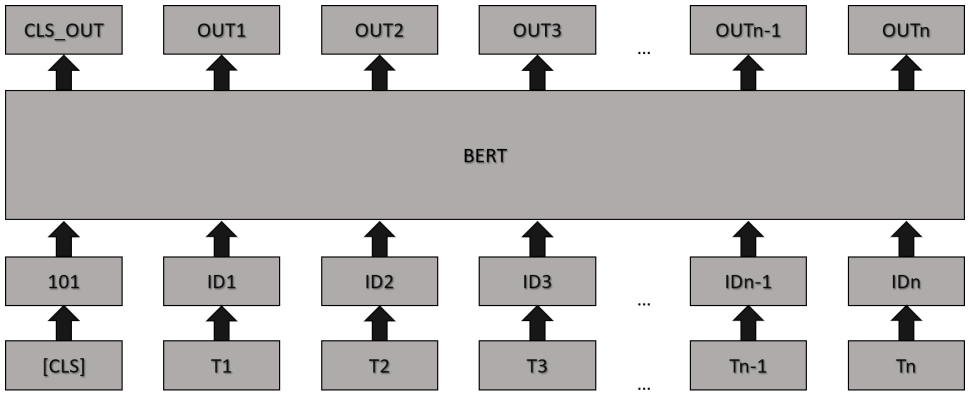


Figura 2.4: Estructura base de Bert

La figura 2.4 representa la estructura que sigue el modelo base de Bert, a partir de esta estructura, se van a desarrollar los diferentes clasificadores del proyecto, entrenándolos utilizando las salidas Out_i obtenidas de Bert.

Precediendo a este tipo de arquitectura, podemos encontrar las redes neuronales recurrentes, este tipo de redes, surgió con el auge del Deep Learning y hasta finales del 2017 se convirtieron en el estándar.

Con la aparición de las redes recurrentes, además comenzaron a surgir las técnicas de embedding, estas, consisten en asignar a cada palabra un vector numérico que representa la información semántica.

CAPÍTULO 3
Clasificación de Valoraciones

3.1 Estado del arte para el Análisis de Valoraciones

El lenguaje humano puede ser una gran fuente de información.

Las palabras, pueden definir el estado de ánimo, mental o incluso los sentimientos de una persona [6].

Las redes sociales, cada vez son una fuente más grande de este tipo de información.

El objetivo del análisis de sentimientos, es extraer la información más subjetiva, por lo general connotaciones que permiten denotar la polaridad del texto, como pudieran ser los adjetivos cuantitativos indefinidos.

Por lo general, debido a la gran subjetividad, los modelos de análisis de sentimientos no consiguen una precisión muy elevada, esto no quiere decir que el modelo sea incapaz de determinar correctamente la puntuación de las valoraciones, simplemente indica que el modelo interpreta las valoraciones y les asocia una puntuación como lo haría cualquier persona.

Una característica a tener en cuenta sobre los modelos de análisis de sentimientos, es que cada modelo se encuentra ligado al contexto en el que ha sido entrenado, quiere esto decir, que un modelo que ha sido entrenado utilizando un conjunto de datos específico obtendrá resultados acordes mientras analice textos relacionados, en cambio, si el texto no tiene ninguna relación con el conjunto de entrenamiento, los resultados serán más inexactos.

Algunas empresas, han comenzado a implantar este tipo de sistemas para automatizar la tarea de extracción de información relevante dentro de las valoraciones que reciben, y por lo tanto reducir el tiempo que necesita la empresa para procesarlas [7].

Un ejemplo de ello, puede encontrarse en el Royal Bank de Escocia, donde utilizan técnicas de NLP para extraer de correos, encuestas y conversaciones con clientes y otros centros causas de insatisfacción y corregirlas para ofrecer un servicio más completo[8].

3.2 Motivación

La idea principal es generar un modelo capaz de clasificar valoraciones según sea positivas o negativas y asignarles una puntuación que vaya acorde.

En el proyecto, debido a que la clasificación de valoraciones, es una de las tareas más comunes, se ha desarrollado un modelo con la intención de comparar los resultados con modelos obtenidos de internet.

El pre-procesamiento de los datos, se ha realizado siguiendo las instrucciones de cada modelo, de este modo, se garantiza obtener resultados más fiables.

3.3 Evaluación del Modelo

Para validar el funcionamiento del modelo, se han realizado un par de experimentos con varios conjuntos de datos, el conjunto Google Play Reviews y el conjunto App Reviews.

3.3.1. Google App Reviews

El conjunto utilizado para realizar la primera aproximación del modelo contiene aproximadamente un total de 16.000 entradas formadas por valoraciones realizadas en aplicaciones de la Google Play Store.

El conjunto, se encuentra distribuido por la puntuación de las valoraciones, expresado gráficamente a continuación,

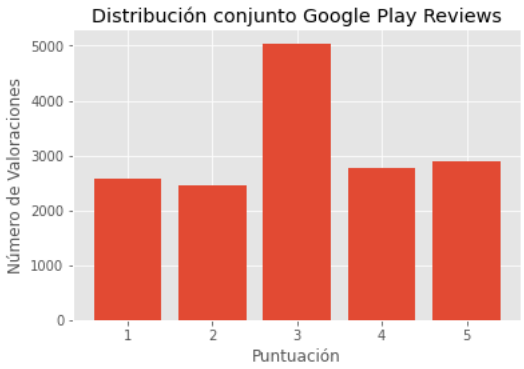


Figura 3.1: Distribución del conjunto Google App Reviews

Como se puede apreciar, a excepción de las valoraciones cuya puntuación P_i equivale a $P_i \in 3$, la distribución se encuentra balanceada.

Por ello y siguiendo las instrucciones del modelo de ejemplo, el conjunto de datos se agrupa en un número de clases $C = 3$, la clase de valoraciones negativas $P_{neg} \in [1, 2]$, las valoraciones neutras $P_{neutra} \in [3]$, y finalmente las valoraciones positivas $P_{pos} \in [4, 5]$.

Figura 3.2: Conjunto Google App Reviews redistribuido

Con la nueva distribución aplicada, el conjunto resultante se encuentra mucho más balanceado.

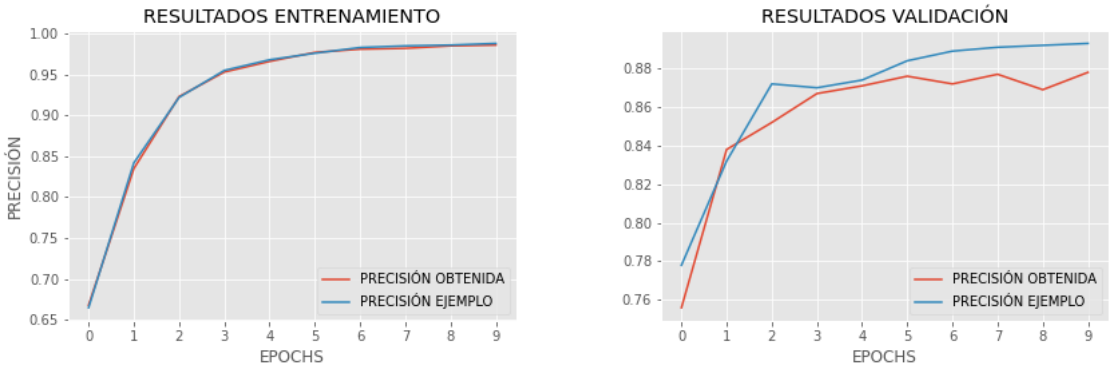


Figura 3.3: Resultados obtenidos Google App Reviews

Como se puede observar en la figura 3.3, la gráfica de entrenamiento, se asemeja a la del ejemplo.

Además, podemos comprobar que la gráfica, es creciente, indicio que permite afirmar que el modelo se está entrenando correctamente.

Por otra parte la gráfica de valoración muestra una pequeña diferencia en los resultados obtenidos.

Para la selección de la mejor Epoch en la etapa de validación, se ha utilizado el modelo de selección por patience.

Finalmente, se obtiene en la etapa de test para el conjunto Google App Reviews un total de 88,19 % de precisión, para este conjunto, se esperaba obtener una precisión cercana a 88,32 %.

3.3.2. App Reviews

El otro conjunto de datos utilizado para comprobar el funcionamiento del modelo diseñado en el proyecto, es un conjunto formado por Valoraciones de Aplicaciones, del mismo

modo que el conjunto previo, se han seguido las indicaciones adjuntas en el modelo de ejemplo, para obtener una comparación lo mas similar posible.

En este caso, nos encontramos con un modelo que contiene aproximadamente 18.000 valoraciones, clasificadas por puntuación,

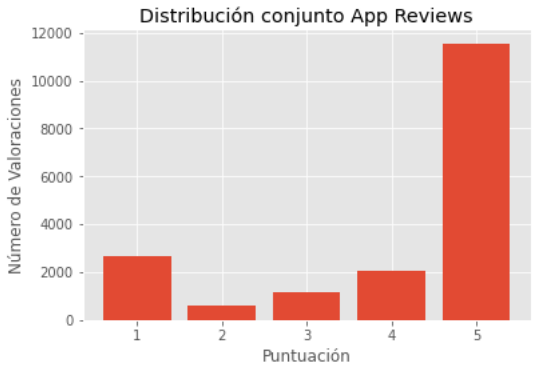


Figura 3.4: Distribución del conjunto App Reviews

Como se puede observar en la figura 3.4, el conjunto App Reviews se encuentra mucho más desbalanceado, para solucionarlo las valoraciones se agrupan en dos clases.

La clase positiva P_{pos}, que contiene las valoraciones más positivas, es decir $P_{pos} \in [5]$ y la clase negativa P_{neg} que contiene el resto de puntuaciones $P_{neg} \in [1, 4]$.

El conjunto redistribuido queda de la siguiente manera,

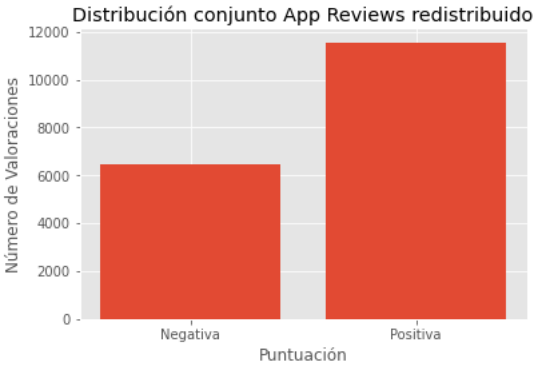


Figura 3.5: Conjunto App Reviews redistribuido

Posteriormente, se entrena de nuevo el modelo desarrollado.

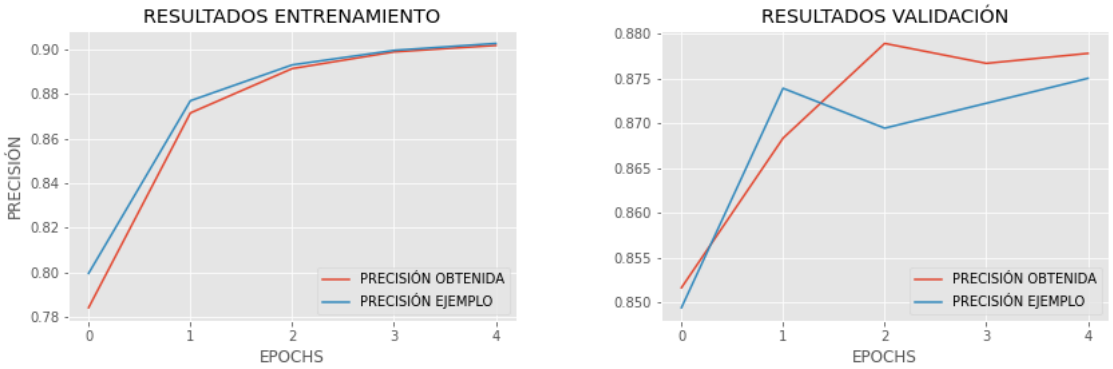


Figura 3.6: Resultados obtenidos App Reviews

Como se puede comprobar en la figura 3.6, la gráfica de entrenamiento se comporta de la misma manera que en con el conjunto formado por valoraciones de la Google Play, en la etapa de entrenamiento podemos observar como los resultados son constantemente crecientes, mientras que en la validación se puede observar una pequeña variación de resultados.

Finalmente, en la etapa de Test, para el conjunto App Reviews, se ha obtenido una precisión del 87,83 %, la precisión exacta esperada es de 88,11 %, como se puede observar, los resultados son bastante próximos.

3.4 Clasificador de Valoraciones

Tras comprobar que los resultados del modelo desarrollado se aproximan a los resultados obtenidos de ejemplos, se realizan varias modificaciones al modelo para generar un clasificador por puntuación donde para cada valoración, el clasificador es capaz de determinar la puntuación S que mejor se ajusta, cuantificada en el rango $S_i = max[p(1), p(2), ..., p(5)]$

Para entrenar el modelo, se ha utilizado un conjunto de datos obtenido de TripAdvisor[1].

El conjunto se encuentra distribuido por puntuación,

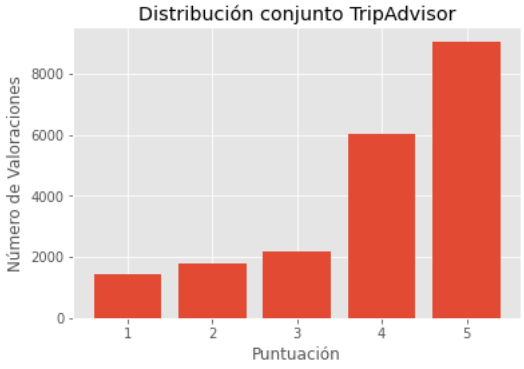


Figura 3.7: Distribución del conjunto TripAdvisor

[1]Link TripAdvisor

A primera vista, se puede observar que existe un desbalanceo en las valoraciones cuya puntuación equivale a $S \in [1, 2, 3]$.

Como primer paso, para adecuar el conjunto de datos, analizamos la longitud de las valoraciones tokenizadas.

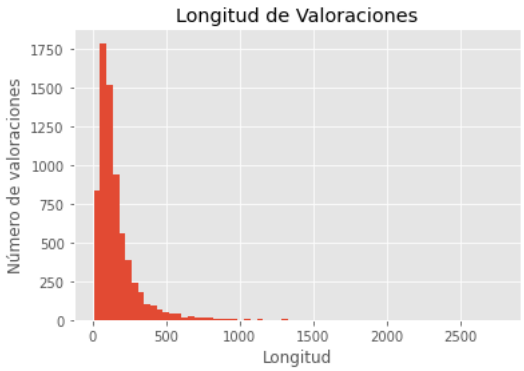


Figura 3.8: Longitud valoraciones tokenizadas

En la figura 3.8 se puede observar como una gran cantidad de valoraciones no contienen más de 500 tokens.

Se eliminan del conjunto las valoraciones cuya longitud es mayor a 500 tokens y se establece como longitud máxima para la tokenización.

Tras adecuar el conjunto de datos, se entrena el modelo realizando los ajustes oportunos, para que el modelo sea capaz de clasificar las valoraciones según su puntuación.

Tras realizar el entrenamiento, los resultados de valoración han sido los siguientes,

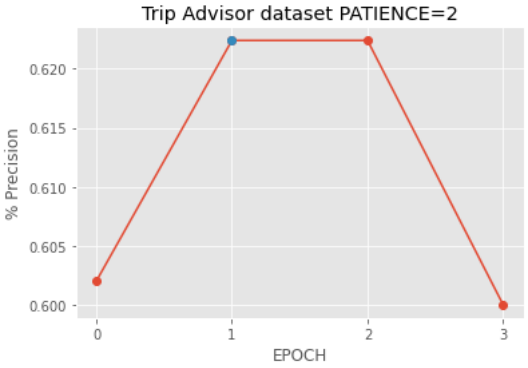


Figura 3.9: Resultados validación TripAdvisor

Finalmente, partiendo de la epoch con mayor porcentaje en validación, se realiza la etapa de test, obteniendo un porcentaje de precisión igual al 62,24 % en el modelo final.

CAPÍTULO 4
Clasificación de Noticias

4.1 Estado del arte para la clasificación de Noticias

Este tipo de modelos se utiliza para ofertar servicios personalizados, dentro de este campo, encontramos el proyecto Mercurio [9], este proyecto, utiliza técnicas de procesamiento del lenguaje natural sumado a técnicas de modelado de usuario para ofrecer información más ajustada.

La clasificación de noticias, funciona muy bien con modelos de filtrado, ya que permite a los usuarios, obtener la información lo más concreta posible, evitando de este modo, perder mucho tiempo realizando extensas búsquedas en la web.

Otra de las utilidades que se atribuyen a este tipo de modelos, es la detección de las noticias falsas, conocidas comúnmente como "fake news", con el incremento de información publicada en internet, este tipo de tareas de detección automática de noticias falsas, es cada vez más demandado.

4.2 Motivación

La idea de generar un modelo de clasificación de noticias utilizando Bert, surgió como idea tras revisar el funcionamiento del modelo usado por la clasificación de sentimientos.

Este modelo parte de la base que se ha utilizado en la clasificación de valoraciones, utiliza Bert, para obtener la información subjetiva y con ello, diferenciar las categorías de los artículos.

4.3 Evaluación del modelo

4.3.1. Conjunto de noticias

Para realizar la tarea del clasificador de noticias, se ha utilizado un conjunto de datos formado por aproximadamente 200000 artículos, distribuidos en 41 clases diferentes.

La distribución que sigue el conjunto es la que sigue,

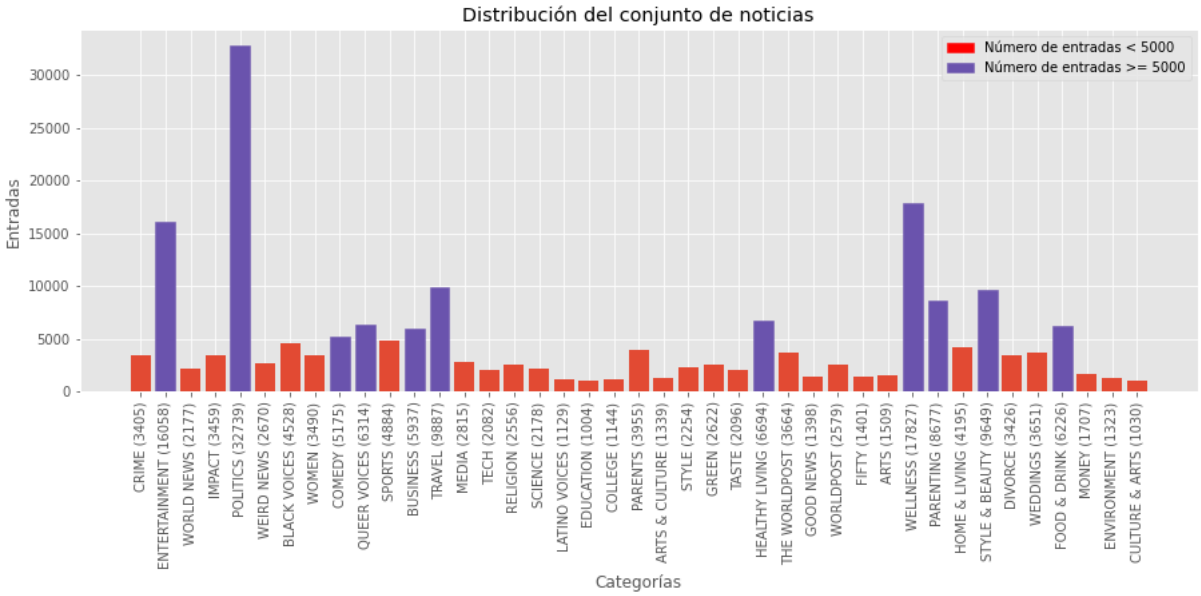


Figura 4.1: Distribución del conjunto de noticias

Como se puede apreciar en la figura 4.1, el conjunto se encuentra muy desbalanceado, la cantidad de entradas varía en gran medida entre las diferentes categorías.

Para solucionar este problema, se generan dos conjuntos diferentes, el primer conjunto, seleccionando de las 41 categorías distintas el valor mínimo de entradas (1.030), para el segundo conjunto, se seleccionan las 11 categorías que tienen una cantidad de entradas igual o superior a 5.000.

Para ambos conjuntos se han analizado la longitud de las entradas tokenizadas, podemos encontrar para el conjunto de 41 categorías una longitud máxima de 64 tokens,

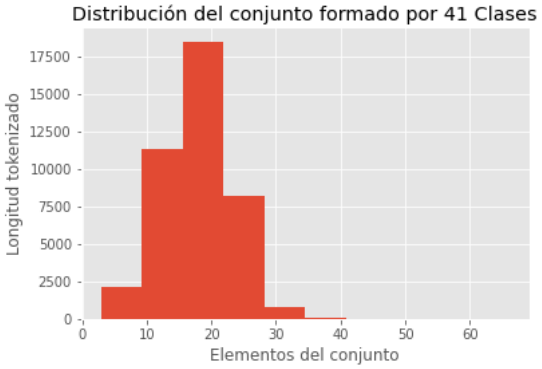


Figura 4.2: Longitud cabeceras tokenizadas 41 categorías

Mientras que para el conjunto formado por 11 categorías el resultado ha sido de una longitud máxima de 50 tokens,

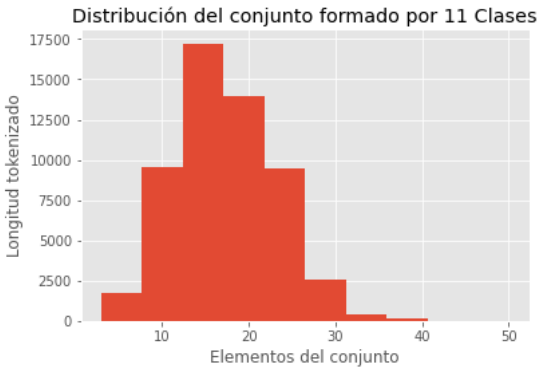


Figura 4.3: Longitud cabeceras tokenizadas 11 categorías

Una vez obtenidas las longitudes máximas de ambos subconjuntos, se entrenan los modelos, los resultados con $PATIENCE = 2$ son los siguientes,

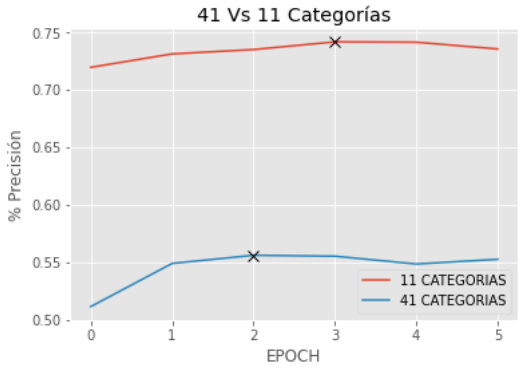


Figura 4.4: Comparativa resultados distribuciones 11 categorías y 41 categorías

Como se puede comprobar en la figura 4.4, el modelo entrenado con 11 categorías ha obtenido mejores resultados en validación.

Finalmente, se realiza la etapa de test, y los resultados de precisión obtenidos son los siguientes,

	41 Categorías	11 Categorías
Precisión	56.30 %	76.12 %

Como puede observar, el conjunto que contiene 11 categorías obtiene mejores resultados que el conjunto que contiene 41 categorías, esto era de esperar, debido a que para 11 categorías, se han podido analizar más muestras y al tener un número más reducido de clases, existe menor dispersión.

CAPÍTULO 5
Marco experimental

5.1 Tokenizador de Bert

e datos,La tokenización es un proceso que se encarga de dividir un texto en elementos más sencillos con significado propio, además se realiza un proceso de limpieza, eliminando del texto los carácteres que no interesan como signos de puntuación o símbolos especiales.

Bert incluye dos modelos de tokenización,

- **BERT Uncased**: Se pasan a minúsculas todas las palabras y se eliminan todos los acentos
- **BERT Cased**: Se realiza la tokenización manteniendo los carácteres en mayúscula y los acentos

Para preparar el texto para ser analizado por Bert, en la etapa de tokenización se añaden algunos tokens especiales, que además se mencionan a lo largo de la memoria y se encuentran explicados a continuación,

- **Token [CLS]:** Token que indica el comienzo del texto, y tras realizar el análisis es el token en el que se concentra la mayor cantidad de información relacionada con el contexto.
- **Token [SEP]:** Token que indica la separación entre textos.
- **Token [MASK]:** Token especial que utiliza Bert para realizar predicciones, utilizado para encontrar las palabras que aportan información relevante al texto.
- **Token [PAD]:** Token de relleno, utilizado en la etapa de tokenización del texto.

5.2 Experimento a partir de los resultados de Bert

5.2.1. Introducción

Para realizar este experimento, se realizan varias operaciones a las salidas que se obtienen al procesar el texto con Bert, se utilizan principalmente dos operaciones, ya que los resultados oscilan en un rango [-1,1], estas operaciones se definen como:

$$Suma = \sum_{i=1}^{N} Out_i$$

$$MediaAritmtica = \left(\frac{1}{N} * \sum_{i=1}^{N} Out_i \right)$$

$$Producto = \prod_{i=1}^{N} Out_i$$

Donde N representa el número de palabras tokenizadas que forman el texto y Out hace referencia a los resultados obtenidos en la salida de Bert para cada uno de los tokens i-ésimos.

5.2.2. Aplicación en el proyecto

El experimento se ha realizado tanto en el clasificador de valoraciones como en el clasificador de noticias, para comparar con los resultados obtenidos en el experimento con los resultados originales.

5.2.3. Estructura del experimento

Para realizar el experimento, se han realizado dos variantes, en la primera variante, anulamos el valor en la salida del token [CLS], es decir, $0 * \text{Out}_{\text{Cls}}$ la nueva estructura que sigue el modelo se puede ver representada en la figura 5.1,

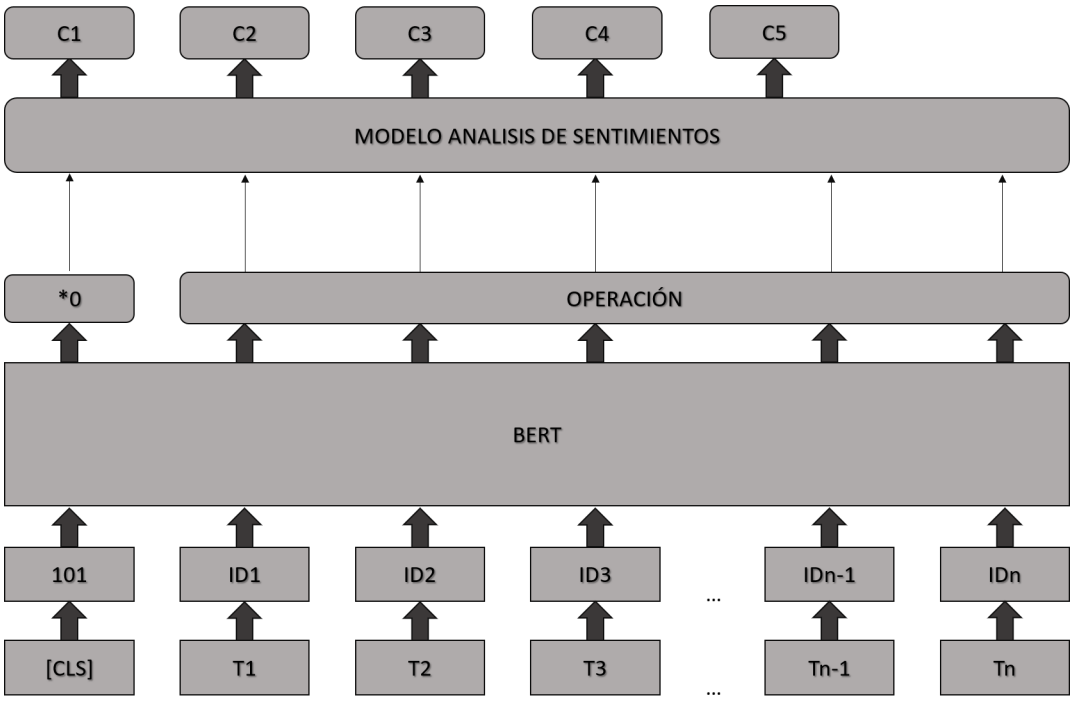


Figura 5.1: Estructura del experimento sin el token [CLS]

Para la siguiente variante, se ha incluido el resultado obtenido en el token [CLS], como podemos ver representado en la figura 5.2,

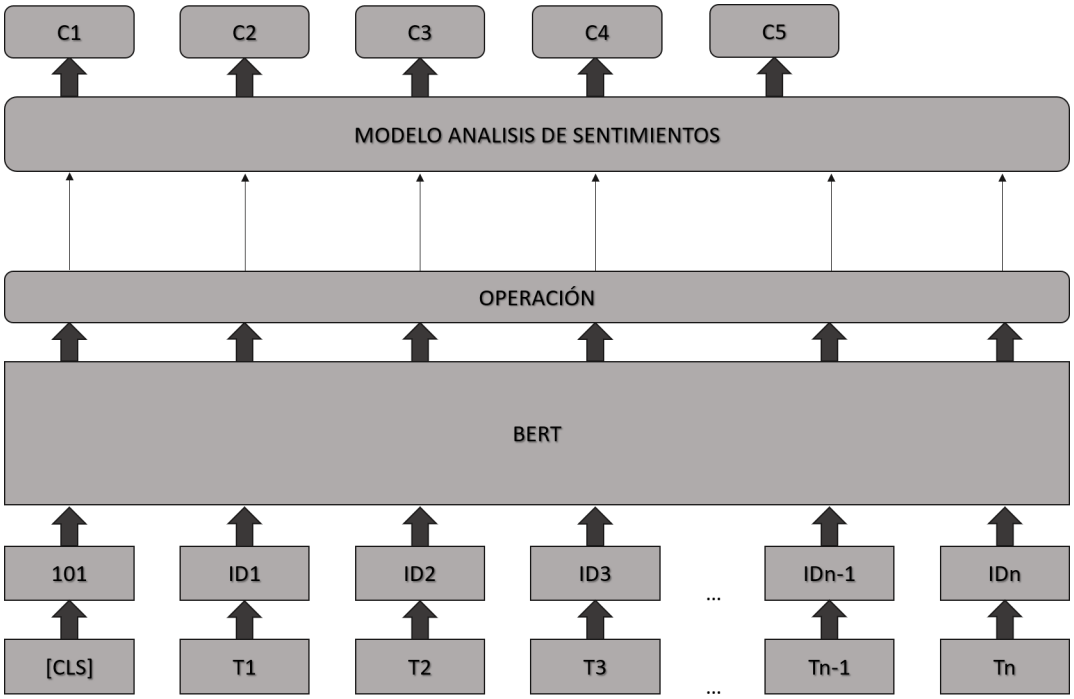


Figura 5.2: Estructura del experimento con el token [CLS]

A la hora de realizar ambas variantes, se han tenido en cuenta los tokens especiales de relleno representados con el símbolo [PAD], los resultados obtenidos para estos tokens, han sido anulados para que no influyan en la clasificación, como se puede ver representado en la figura 5.3,

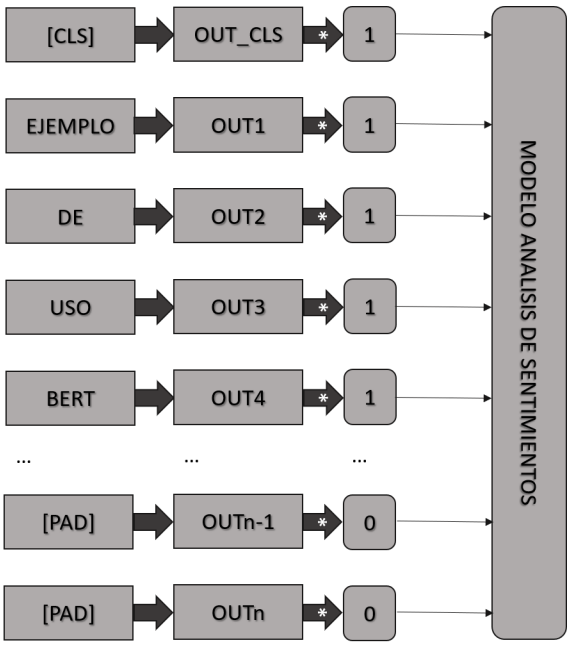


Figura 5.3: Tratamiento de los tokens [PAD]

5.2.4. Resultados para el clasificador de valoraciones

Una vez presentadas las distintas operaciones, se entrenan ambos modelos y se comparan los resultados,

La siguiente gráfica muestra los resultados obtenidos sin el token [CLS],

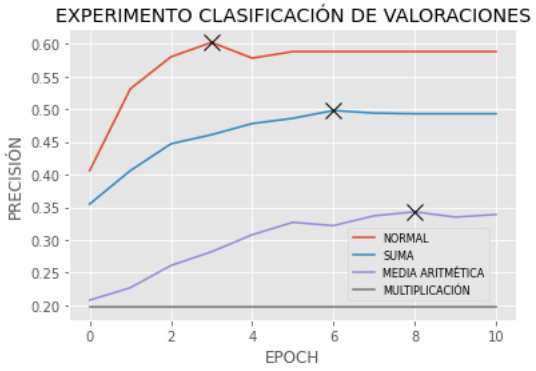


Figura 5.4: Resultados experimento análisis de valoraciones sin [CLS]

Como se puede observar, el modelo al que se le aplica la operación de suma obtiene los mejores resultados, aún así no es suficiente para mejorar los resultados del modelo base.

La media aritmética, obtiene una precisión muy baja.

Finalmente, como era de esperar, la multiplicación obtiene resultados no fiables, debido a que los resultados obtenidos de Bert, están comprendidos entre los valores $Out_i = [-1, 1]$ aprox.

Los resultados del modelo al que se le ha añadido el token [CLS], son los siguientes,

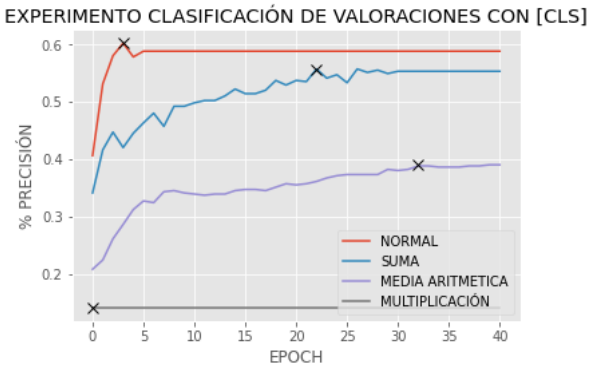


Figura 5.5: Resultados experimento análisis de valoraciones con [CLS]

Los resultados obtenidos son mejores que los del modelo sin el tokens [CLS], pero aún así no mejoran los resultados del modelo base.

La interpretación de los resultados se asemeja a la del modelo sin el token [CLS].

La suma, obtiene los mejores resultados sin mejorar el modelo base, la media aritmética obtiene resultados insuficientes y la multiplicación muestra resultados de un modelo poco fiable.

5.2.5. Resultados para el clasificador de noticias

Para el modelo del clasificador de noticias vamos a utilizar las mismas técnicas empleadas en el experimento realizado al análisis de valoraciones, para observar si existe alguna mejoría en este modelo.

Se han aplicado los dos modelos descritos, el modelo aplicando los resultados obtenidos en el token [CLS] y el modelo en el que no se aplican.

Los resultados obtenidos en el modelo sin utilizar el token [CLS], son los siguientes,

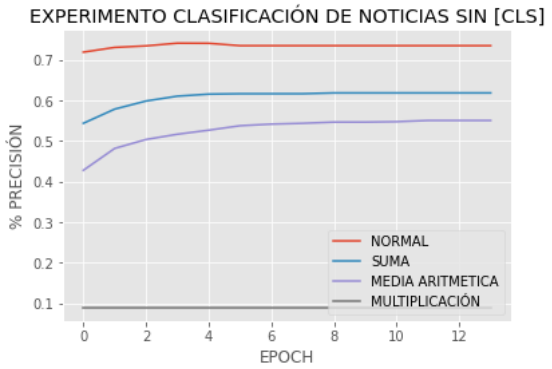


Figura 5.6: Resultados experimento clasificador de noticias sin [CLS]

Al igual que en el experimento realizado con el análisis de valoraciones, no se obtienen resultados que mejoren el modelo base.

Las operaciones aplicadas se comportan de igual manera, obteniendo que para la suma se generan los mejores resultados de entre las demás operaciones sin mejorar los resultados obtenidos por el modelo base, mientras que para la media aritmética, se obtiene resultados insuficientes y para la multiplicación se observa que es un modelo incongruente.

Los resultados obtenidos para el modelo al que se le añade e token [CLS], son los siguientes,

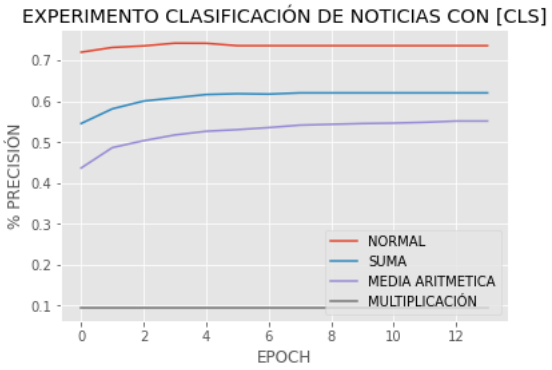


Figura 5.7: Resultados experimento clasificación de noticias con [CLS]

Para el caso del clasificador de noticias, la diferencia entre los modelos no es muy notable.

Las operaciones aplicadas en el modelo que utiliza el token [CLS] en las operaciones siguen el mismo comportamiento que las operaciones aplicadas en el modelo que no utiliza el token [CLS].

5.3 Experimento K-BERT

5.3.1. Introducción

Este modelo se asemeja al modelo K-BERT [10], el cual utiliza un KG (Knowledge Graph) también conocido como base de conocimiento.

A partir de esta base de conocimiento, se añade información adicional al texto de entrada, permitiendo hacer hincapié en las características más relevantes de una palabra

5.3.2. Aplicación en el proyecto

Este modelo K-BERT, se ha desarrollado en la tarea de clasificación de valoraciones, utilizando una base de conocimiento capaz de identificar los adjetivos calificativos del texto e indicar dentro del texto si representan un sentimiento positivo o un sentimiento negativo.

5.3.3. Desarrollo de Base de conocimiento

Para desarrollar la base de conocimiento del experimento, se obtuvo una lista generada por ordenador con una gran cantidad de adjetivos, de los cuales inicialmente, se clasificaron aproximadamente 300 adjetivos en sentimientos positivos y negativos.

Con el conjunto de 300 adjetivos clasificados manualmente, se entrenó un modelo utilizando Bert para clasificar los restantes adjetivos, obteniendo un 77.50 % de acierto para test.

Inicialmente, la idea fue de obtener un modelo con mínimo un 80.00 % de acierto en testing, por ello se clasificaron manualmente aproximadamente otros 200 adjetivos.

Con el nuevo conjunto formado por 500 adjetivos clasificados manualmente, se volvió a entrenar el modelo obteniendo un total de un 87.50 % de acierto en test, y por lo tanto, el modelo fue utilizado para clasificar automáticamente el resto del conjunto (aprox. 4000 adjetivos).

A la hora de desarrollar el experimento con el clasificador de valoraciones, en las entradas, se añadieron un total de 83.055 elementos indicando si el adjetivo que le precede expresa un sentimiento negativo o por lo contrario expresa un sentimiento positivo.

5.3.4. Resultados obtenidos

Una vez completa la base de conocimiento, se realiza el preprocesamiento del conjunto TripAdvisor, y se entrena de nuevo el modelo.

En la etapa de test, en el modelo original, se ha obtenido 51,63 % de precisión, utilizando las técnicas de K-BERT, la precisión obtenida es de 60,00 %.

5.4 Conclusiones

Como se ha podido comprobar en los experimentos, la salida del token [CLS] abstrae de mejor manera el ámbito del texto a analizar, permitiendo generar modelos con mayor porcentaje de acierto.

Por otro lado, como se ha podido observar en el modelo generado K-BERT, el incluir información adicional sobre la polaridad del texto, permite obtener mejores resultados en la tarea de clasificación de valoraciones.

CAPÍTULO 6
Desarrollo del demostrador

6.1 Introducción

Para otorgar una mayor consistencia al proyecto, se ha diseñado una aplicación de escritorio donde se reúnen los modelos que se han desarrollado dentro del proyecto.

Para el desarrollo de la interfaz gráfica, se ha utilizado el software QT Designer, para la generación de código en Python, se ha utilizado PyQT5, finalmente, para generar los checkpoints utilizados de las redes neuronales, se ha utilizado la librería Pytorch.

6.2 Clasificadores

La aplicación, se compone de una pantalla principal con un selector de utilidades en la parte inferior, donde se puede seleccionar el modelo de clasificación, la aplicación contiene las tareas de clasificación de valoraciones y de clasificación de noticias a la que también se ha incluido la predicción de mascaras representadas con el token [MASK].

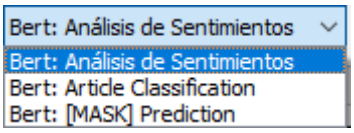


Figura 6.1: Selección de modelo

Adicionalmente, podemos encontrar varios campos de texto, uno de entrada, donde se presenta el texto a analizar y otro campo no modificable de salida, donde se presentan los resultados del análisis.

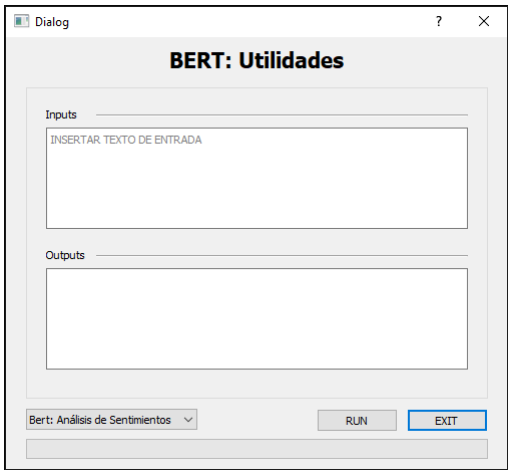
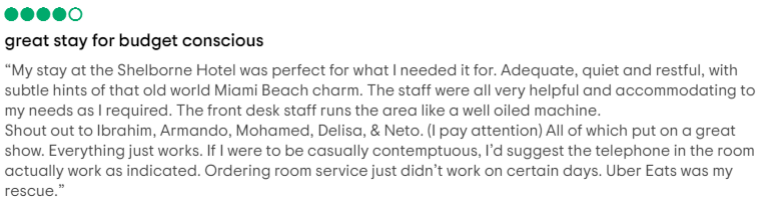


Figura 6.2: Pantalla de inicio

Para cada modelo de clasificador, se ha exportado un checkpoint de las redes neuronales ya entrenadas, utilizando para ello, una de las utilidades que presenta la librería PyTorch, estas redes se cargan al iniciar la aplicación.

Como ejemplo, vamos a utilizar la siguiente valoración obtenida en TripAdvisor, como se puede ver, tiene una puntuación de cuatro estrellas.



great stay for budget conscious

"My stay at the Shelborne Hotel was perfect for what I needed it for. Adequate, quiet and restful, with subtle hints of that old world Miami Beach charm. The staff were all very helpful and accommodating to my needs as I required. The front desk staff runs the area like a well oiled machine.
Shout out to Ibrahim, Armando, Mohamed, Delisa, & Neto. (I pay attention) All of which put on a great show. Everything just works. If I were to be casually contemptuous, I'd suggest the telephone in the room actually work as indicated. Ordering room service just didn't work on certain days. Uber Eats was my rescue."

Figura 6.3: Valoración de TripAdvisor

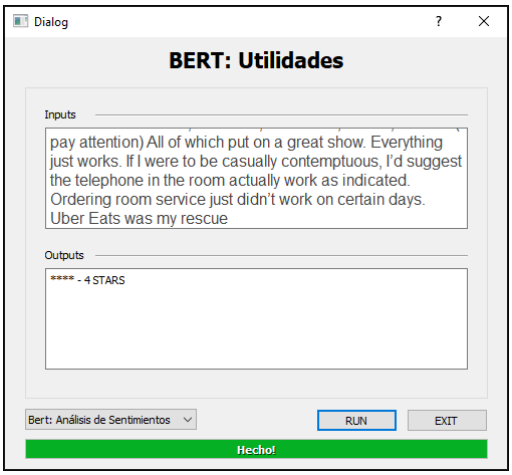


Figura 6.4: Ejemplo de uso del clasificador de valoraciones

En la figura 6.4, podemos observar que en este caso, el resultado obtenido coincide con la valoración de la puntuación.

El siguiente ejemplo representado en la figura 6.5, muestra un ejemplo de uso para el clasificador de noticias.

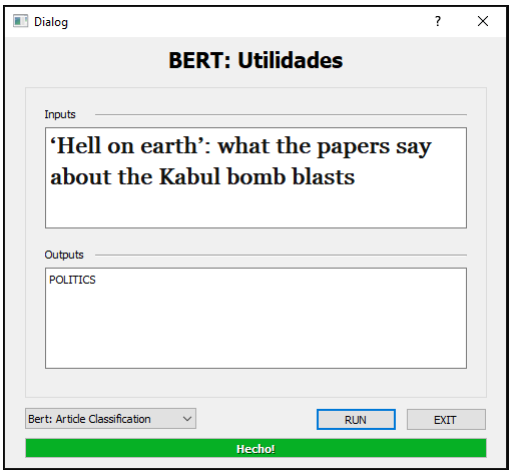


Figura 6.5: Ejemplo de uso del clasificador de noticias

Adicionalmente a las utilidades vistas en el proyecto, se ha añadido una utilidad extra denominada predicción de palabras enmascaradas.

Esta predicción no tiene una utilidad muy aplicable a un problema práctico, pero si es importante ver su funcionamiento y con ello, comprender mejor como Bert consigue relacionar las palabras con el contexto en el que se encuentran.

Bert es un modelo que utiliza un modelo de análisis bidireccional, entrando en detalle, utiliza un método conocido como análisis bidireccional profundo, para realizar este tipo de análisis del texto, en la etapa de entrenamiento de Bert, se enmascaran utilizando un token especial representado como [MASK] y posteriormente se realiza una predicción de estos tokens.

Para realizar la predicción el texto de entrada se proporciona un 80 % de las veces con el token [MASK] sustituyendo la palabra a predecir, otro 10 % con un token aleatorio y el otro 10 % restante con la palabra correcta.

Esto se utiliza para determinar que tokens aportan información al contexto, si el modelo es capaz de predecir la palabra correcta, eso quiere decir, que ese token aporta información relevante al texto.

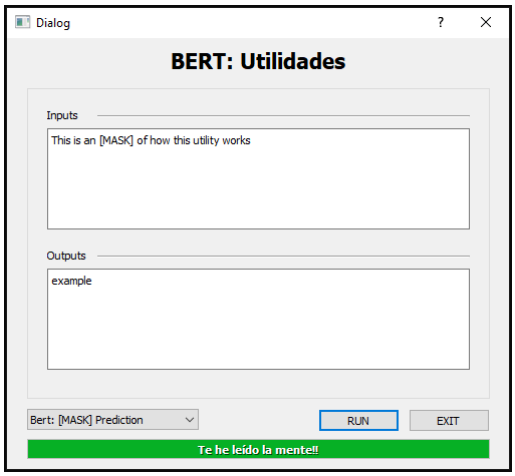


Figura 6.6: Ejemplo de uso del predictor de máscaras

Como podemos ver en la figura 6.6, tenemos un ejemplo en el que se predice el token [MASK] obteniendo como token resultante, la palabra “example”.

6.3 Optimizaciones

Al utilizar modelos con un tamaño muy elevado, uno de los aspectos que se han tenido en cuenta a la hora de diseñar la aplicación, han sido los tiempos de carga.

Los modelos de Bert, se descargan de internet, y se almacenan en la caché del ordenador, con lo que la carga inicial suele ser un poco más elevada.

Para crear y cargar los modelos diseñados en el proyecto, se han utilizado técnicas de multiprocesamiento, de esta manera, se consigue paralelizar las cargas de los distintos modelos y por lo tanto reducir el coste temporal.

El tiempo sin utilizar técnicas de paralelismo, viene definido por la suma de todos los modelos t_m a ser cargados, matemáticamente el coste temporal $T_s$ecuencial para N modelos, se expresa de la siguiente manera,

$$T_{secuencial} = (t_{m_1} + t_{m_2} + ... + t_{m_N}) \equiv \sum_{i=1}^{N} t_{m_i}$$

El tiempo utilizando técnicas de paralelismo, viene definido por el coste del modelo t_m que más tarda en ser cargado, matemáticamente, el coste temporal $T_p$aralelo para N modelos, se expera como sigue,

$$T_{paralelo} = MAX(t_{m_1}, t_{m_2}, ..., t_{m_N})$$

Tras implementar las optimizaciones, se recopilan los tiempos de carga de la aplicación, utilizando técnicas de Threading, Multiprocessing y sin ellas, teniendo ya descargado en la caché el modelo de Bert, los resultados expresados en segundos que se han obtenido son los siguientes,

	Tiempo 1	Tiempo 2	Tiempo 3	Tiempo 4
Base	23.39	22.46	22.40	24.12
Threading	11.21	13.20	11.10	12.54
Multiprocessing	12.34	13.11	11.21	10.28

Como podemos observar en la tabla, las técnicas de Threading y Multiprocessing obtienen resultados similares, mejorando los obtenidos por la aplicación base.

Para el paralelismo, se decidió utilizar multiprocessing creando para cada modelo un proceso de carga independiente y devolviendo los resultados al proceso que los creó mediante una cola compartida.

CAPÍTULO 7
Conclusiones

7.1 Conclusiones del proyecto

A lo largo del proyecto, se ha presentado una red neuronal entrenada por Google denominada Bert y partiendo de ella, se han generado varios modelos de clasificación de texto.

En el primer capítulo, se han revisado los objetivos del proyecto, las expectativas y se han introducido algunos de los conceptos básicos que se mencionan a lo largo del proyecto.

En el segundo capítulo, se presenta la red Bert, su historia y su arquitectura, para poder entender como funciona.

En el tercer capítulo, se implementa un clasificador de valoraciones para comprobar el correcto funcionamiento del modelo implementado en el proyecto, para ello se comparan los resultados obtenidos con resultados de ejemplo.

En el cuarto capítulo, partiendo del modelo desarrollado para la clasificación de valoraciones, se desarrolla un modelo de clasificación de noticias.

En el quinto capítulo, se presentan algunos de los procesos que realiza Bert, como la tokenización, para introducir los experimentos que se van a realizar, presentando los resultados obtenidos.

Por último, el sexto capítulo, introduce el demostrador desarrollado, en la que se incluyen los modelos base de los distintos clasificadores desarrollados, en este capítulo, se presentan las optimizaciones temporales que se han implementado dentro del proyecto.

7.2 Trabajos Futuros

Como se ha revisado en el proyecto, Bert es una herramienta muy útil para las tareas relacionadas con el análisis de texto, permite relacionar las palabras con el contexto que las rodea, gracias a esto, es capaz de generar una representación consistente del texto que está analizando.

Los modelos de Bert, tienen mucha aplicación hoy en día, debido a la gran cantidad de publicaciones que se realizan en Internet, siguiendo la línea del análisis de texto, se podrían analizar comentarios en redes sociales para detectar comportamientos no deseados, como también se podrían analizar noticias para detectar que noticias son falsas.

Estos últimos son ejemplos de tareas relacionadas con las implementadas en el proyecto, que partiendo de los modelos y los resultados obtenidos sirven de base para su desarrollo.

7.3 Relación con los estudios cursados

En la realización del proyecto, han resultado muy útiles muchos de los conceptos adquiridos en la Universidad Politécnica de Valencia, muchos de ellos han servido para entender el funcionamiento de las redes neuronales e implementar los modelos desarrollados en el proyecto, muchos de estos conocimientos han sido adquiridos en asignaturas de la rama de computación como son Percepción (PER) y Aprendizaje Automático (APR).

Adicionalmente, para el desarrollo e implementación de las optimizaciones, conocimientos adquiridos en asignaturas como Computación paralela (CPA) y Concurrencia y sistemas distribuidos (CSD), han resultado útiles para entender el paralelismo, realizar las comparaciones de tiempo y las diferentes tomas de decisiones que se han realizado a la hora de elegir las mejores optimizaciones.

Para el desarrollo y adecuación del demostrador se han aplicado algunas de las técnicas aprendidas en la asignatura de Interfaces persona computador (IPC).

Muchas de las técnicas relativas a la gestión del proyecto y gestión de tiempos de desarrollo, vienen de asignaturas como Deontología y profesionalismo (DIP) y Gestión de proyectos (GPR).

Finalmente, gracias a los conocimientos de programación obtenidos en asignaturas como Introducción a la programación (IIP), Programación (PRG), Estructuras de datos y algorítmos (EDA) y Sistemas de almacenamiento y recuperación de información (SAR), la programación de los modelos del proyecto ha sido posible.

Bibliografía

[1] Dr. Dataman, 2018, Looking into Natural Languaje Processing (NLP). https://medium.com/dataman-in-ai/natural-language-processing-nlp-for-electronic-health-record-ehr-part-i-4cb1d4c2f24b

[2] Pandu Nayak, 2019, Understanding searches better than ever before. https://blog.google/products/search/search-language-understanding-bert/

[3] Rafael Lucca, ¿Qué es BERT y cómo funciona?. https://dxmedia.net/algoritmo-bert-google/

[4] Talamé, L., Cardoso, A., Amor, M. (2019). Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python. In XX Simposio Argentino de Inteligencia Artificial (ASAI 2019)-JAIIO 48 (Salta).

[5] Brownlee, J. (2018). What is the Difference Between a Batch and an Epoch in a Neural Network?. Machine Learning Mastery, 20.

[6] Pennebaker, J. W., Mehl, M. R., Niederhoffer, K. G. (2003). Psychological aspects of natural language use: Our words, our selves. Annual review of psychology, 54(1), 547-577.

[7] Abad Fernández, D. (2016). Técnicas estadísticas aplicadas a redes rociales y análisis de sentimiento: el caso Volkswagen.

[8] Gallego-Gomez, C., De-Pablos-Heredero, C. (2020). Artificial Intelligence as an Enabling Tool for the Development of Dynamic Capabilities in the Banking Industry. International Journal of Enterprise Information Systems (IJEIS), 16(3), 20-33.

[9] Díaz, A., Gervás, P., Gómez, J. M., García, A., de Buenaga, M., Chacón, I., ... Acero, I. (2000). Mercurio: un servicio personalizado de noticias basado en técnicas de clasificación de texto y modelado de usuario. Procesamiento del Lenguaje Natural, 26.

[10] Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., Wang, P. (2020, April). K-bert: Enabling language representation with knowledge graph. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 03, pp. 2901-2908).

[11] Rémi Louf, 2019, Encoder-decoders in Transformers: a hybrid pre-trained architecture for seq2seq. https://medium.com/huggingface/encoder-decoders-in-transformers-a-hybrid-pre-trained-architecture-for-seq2seq-af4d7bf14bb8

[12] Guardiola González, C. (2020). Clasificador de textos mediante técnicas de aprendizaje automático (Doctoral dissertation, Universitat Politècnica de València).

[13] Álvarez-Daza, N., Pico-Valencia, P., Holgado-Terriza, J. A. (2021). Detección de Noticias Falsas en Redes Sociales Basada en Aprendizaje Automático y Profundo: Una Breve Revisión Sistemática. Revista Ibérica de Sistemas e Tecnologias de Informação, (E41), 632-645.

[14] Sun, C., Qiu, X., Xu, Y., Huang, X. (2019, October). How to fine-tune bert for text classification?. In China National Conference on Chinese Computational Linguistics (pp. 194-206). Springer, Cham.

APÉNDICE A

Modelo base para el desarrollo de los clasificadores

Para la implementación de los diversos clasificadores que se han ido revisando en el proyecto, se diseño un modelo base del que parten las diferentes variantes.

El código empleado en el modelo base, es el siguiente,

```
class BERT(nn.Module):

    def __init__(self, bert_model, dropOut, numClases, max_len):
        super(BERTArticleClassificator, self).__init__()
        self.bert = BertModel.from_pretrained(bert_model)
        self.tokenizer = BertTokenizer.from_pretrained(bert_model)
        self.drop = nn.Dropout(p=dropOut)
        self.linear = nn.Linear(self.bert.config.hidden_size, numClases)
        self.max_len = max_len

    def forward(self, review, bar):
        encoding = self.tokenizer.encode_plus(
            review,
            max_length=self.max_len,
            truncation=True,
            add_special_tokens=True,
            return_token_type_ids=False,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt'
        )

        outputs, cls_output = self.bert(
            input_ids=encoding['input_ids'].flatten().unsqueeze(0),
            attention_mask=encoding['attention_mask'].flatten().unsqueeze(0),
            return_dict=False
        )

        drop_out = self.drop(cls_output)
        output = self.linear(drop_out)
```

```
        return output
```

En la inicialización de la clase BERT (init), se carga el modelo de tokenización y el modelo Bert a utilizar, para ello se hacen uso de las librerías BertModel y BertTokenizer del paquete Transformer, de las cuales se emplea la función from_pretrained que se encarga de cargar los modelos, se especifican e inicializan parámetros como el DropOut, la longitud máxima y la configuración de la capa oculta.

La función forward, se encarga de entrenar la red, tomando como entrada un texto que es tokenizado, obteniendo los vectores input_ids (contiene el id de las palabras) y attention_mask (indica si las palabras tienen relevancia), estos vectores, se utilizan para entrenar la red neuronal.

APÉNDICE B

Procesos internos en las redes transformer

Como se ha mencionado en el apartado 2.3, las redes transformer utilizan una estructura Codificador-Decodificador.

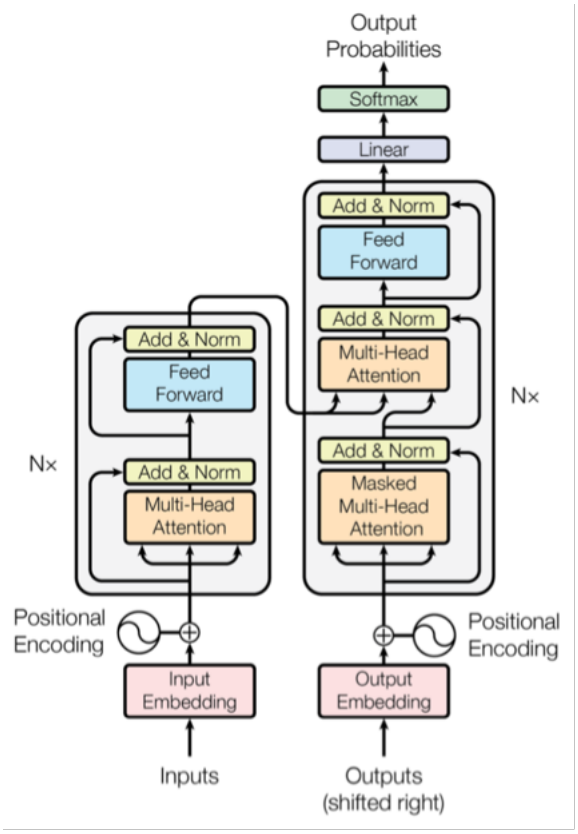


Figura B.1: Estructura CodificadorDecodificador [11]

En la entrada de codificación, podemos encontrar varios procesos, inicialmente, el texto es representado como un conjunto de vectores de números reales, esta técnica es conocida como embedding

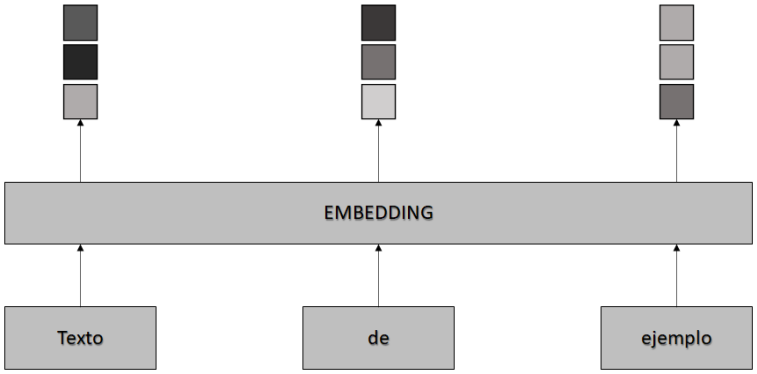


Figura B.2: Ejemplo de embedding

Las redes transformer son capaces de analizar el texto de entrada de manera paralela, para lograr este análisis en paralelo, se le añaden a los vectores de embedding las posiciones relativas de cada token como podemos observar en la figura B.2,

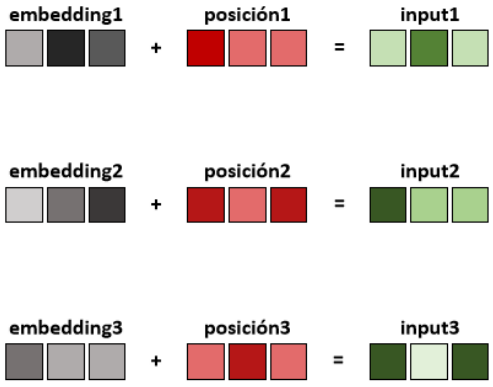


Figura B.3: Embedding posicional

Una vez obtenidos los vectores de entrada representados en la figura B.2, como vectores input, se utilizan en la capa denominada como Bloque atencional, en esta capa, se utilizan tres redes neuronales diferentes denominadas, Query, Keys y Values, de los cuales obtenemos los vectores de codificación como podemos ver en la figura B.3,

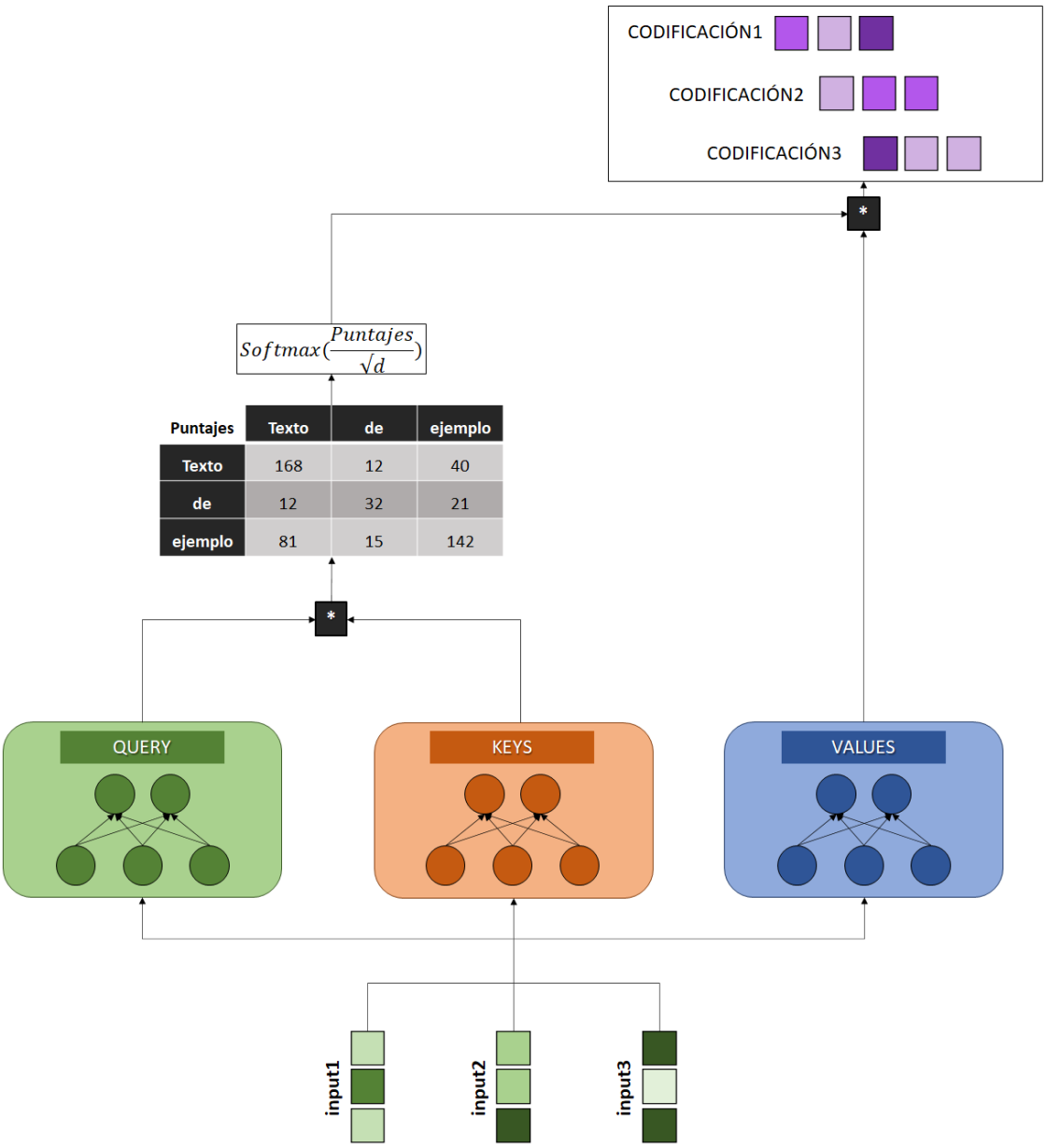


Figura B.4: Estructura del modelo

Incialmente, se comparan la similitud entre el vector Query y el vector Keys obteniendo una matriz de puntuaciones.

Esta matriz se normaliza y se le aplica la función Softmax, para posteriormente, ponderar el vector Values y obtener de esta manera los vectores de codificación del contexto.

Las operaciones vienen expresadas por la siguiente fórmula;

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right) * V.$$

Donde los resultados de las redes Query, Keys y Values vienen representados con la notación $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ respectivamente, y donde $(\mathbf{d})$ expresa la dimensión del texto.

Para analizar las relaciones de palabras dentro del texto de manera más profunda, se utilizan varias capas del bloque atencional, matemáticamente se puede representar de la siguiente manera,

$$\phi_{MH}(Q, V, K) = [\theta_1; \theta_2; ...; \theta_N] * W_T$$

Donde ϕ_{MH} representa la función múltiple, mientras que la función individual θ_i se define de la siguiente manera,

$$\theta_i = Attention(QW_{q_i}, KW_{k_i}, VW_{v_i})$$

Tras obtener los vectores de Atención, nos encontramos varios bloques denominados bloques residuales, su principal función es evitar el degradado de los datos, su estructura es la siguiente,

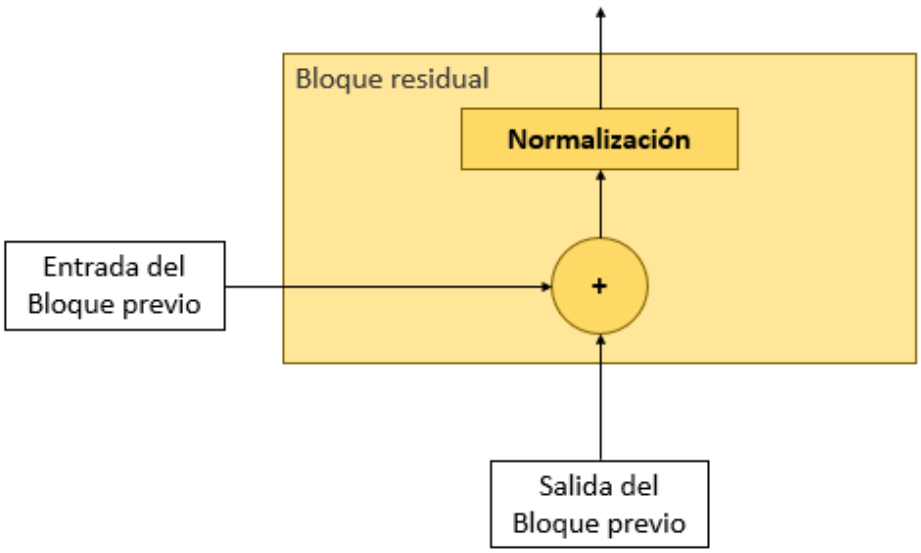


Figura B.5: Bloque residual

Mientras que por otra parte, el decodificador realiza la tarea contraria, recibe de manera secuencial cada una de las palabras y obtiene los resultados correspondientes, sigue el mismo modelo que el codificador pero con un par de variaciones, en la decodificación, tenemos un bloque denominado bloque atencional enmascarado, donde los tokens, son comparados con los que le preceden.

Otra pequeña variación, la podemos encontrar en los bloques atencionales donde los vectores asociados a las redes Query y Keys, se obtienen de parte del codificador, y se utilizan para realizar la ponderación del vector Values, obtenido del bloque de atención enmascarado.

APÉNDICE C

Modelo de selección por Patience

Para seleccionar la mejor epoch en la etapa de validación, se ha implementado una función de selección por patience parametrizado.

Esta función toma como entrada dos valores numéricos, uno que indica la etapa de validación y otro valor que indica que precisión se ha obtenido en dicha etapa de validación.

Expresado en pseudo código, la función tiene la siguiente forma,

```
ETAPAS_PATIENCE = N
valorMaximo = -1
etapaMaximo = 0

def Patience(etapa, valorObtenido):
    if (valorObtenido > valorMaximo):
        valorMaximo = valorObtenido
        etapaMaximo = 0
        return False

    elif (etapa - etapaMaximo >= ETAPAS_PATIENCE):
        print("Stop en etapa {} valor MAX obtenido {}
            en etapa {}'.format(etapa, valorMaximo, etapaMaximo)
        return True
```

Donde el valor ETAPAS_PATIENCE es definido por el usuario para indicar cuantas etapas deben transcurrir hasta que se detenga la validación.

En cada una de las etapas que se encuentra un valor máximo, se guarda el estado del modelo, en caso de que se encuentre un nuevo máximo, se sobre escribe.