Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA
TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

Modelos de Lenguaje DL para la Caracterización de Consejeros Independientes de empresas cotizadas en bolsa

Autor

Juan González Magdalena

Tutores

Benjamín Sahelices Fernández
Fernando Tejerina Gaite

Agradecimientos

Quería dedicar una pequeña sección de este trabajo a las personas que lo han hecho posible. En primer lugar quería dar las gracias a mi tutores Benjamín y Fernando, por su apoyo y dedicación. También quería agradecer a mis amigos y familia, por su apoyo incondicional día a día en estos últimos años. Y por último a los que ya no están, que me cuidan desde allá donde estén.

Gracias.

Resumen

La figura del consejero independiente es una figura fundamental dentro de la gobernanza de las empresas cotizadas en bolsa. La propia CNMV reconoce a esta figura como algo fundamental en grandes momentos de crisis o disrupciones. Por lo cual, un análisis de esta figura es vital para conocer información de las necesidades de estas empresas.

Durante este trabajo se utilizan datos no estructurados extraídos de la web de la CNMV sobre las biografías de los diferentes consejeros independientes a lo largo de unos años. Una parte de estas biografías han sido etiquetados por un experto en economía como es el Dr.Fernando Tejerina Gaite. Donde asigna un valor continuo entre 0 y 1 para los 6 diferentes perfiles principales y 8 subperfiles.

Se plantea la pregunta de clasificar estas biografías de forma automática sin necesidad de un experto humano. Para ello se plantean dos modelos de *deep learning* con arquitecturas LSTM y Transformes para abordar esta clasificación. Posteriormente se tratará de mejorar estos modelos mediante técnicas de *data augmentation*.

Con este trabajo se pretende obtener y comparar como diferentes arquitecturas y técnicas de aumento de datos procesan las biografías de los consejeros independientes con el objetivo de tener una visión general de los diferentes perfiles y subperfiles que buscan estas empresas.

Abstract

The figure of the independent director represents a fundamental aspect of the governance of listed companies. The CNMV itself acknowledges the significance of this figure in times of crisis or disruption. Consequently, an analysis of this figure is essential to gain insight into the needs of these companies.

This paper employs unstructured data extracted from the CNMV website on the biographies of the different independent directors over a number of years. A portion of these biographies have been labelled by an expert in economics, such as Dr. Fernando Tejerina Gaite, who assigns a continuous value between 0 and 1 for the six different main profiles and eight sub-profiles.

The question then arises of classifying these biographies automatically without the need for a human expert. In order to address this classification, two models of deep learning with LSTM and Transforms architectures are proposed. Subsequently, these models will be improved by means of data augmentation techniques.

The objective of this study is to ascertain and contrast the manner in which distinct architectures and data augmentation techniques process biographies of independent directors, thereby gaining insight into the diverse profiles and sub-profiles that these companies are seeking to identify.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Introducción

El aumento de datos en múltiples áreas ha aumentado en los últimos años. Esto representa un hueco potencial a explotar para realizar diversas investigaciones. Más específicamente en el área de Deep Learning se han realizado muchos avances en estos últimos años en diferentes áreas, más específicamente en el sector financiero y corporativo. Donde métodos de *deep learning* pueden obtener datos no estructurados como CVs o biografías permitiendo transformarlo en un análisis cualitativo de perfiles.

Más en profundidad trataremos en el contexto de las empresas cotizadas en el IBEX 35. Donde los directores pertenecientes a estos consejos representan una fracción significativa de la capacidad de toma de decisiones en los consejos de administración. Cada año, el Informe Anual de Gobierno Corporativo se publica en el sitio web de la Comisión Nacional del Mercado de Valores (CNMV). Parte de este informe se compone de los CV de los diferentes directivos, por lo que su formato es de texto libre no estructurado, lo que dificulta enormemente su análisis.

Para modelar y analizar el funcionamiento de estas empresas y entender las razones detrás de sus decisiones estratégicas, es muy importante conocer el trasfondo y la carrera profesional de los directores. Sin embargo, el análisis de esta información debe hacerse de manera manual y, por lo tanto, requiere grandes cantidades de recursos dados el gran número de personas, empresas y años involucrados. Por ello el Dr.Fernando Tejerina Gaite ha realizado un etiquetado manual de más de 1000 biografías, dando una puntuación del 0 al 1 a 6 perfiles principales (F, E/C, A/T/A, L, P, Ac) y 8 subperfiles (FB, FnB, CEO, E/C_E , E/C_M , E/C_RH , E/C_I , E/C_F) con un solo dígito decimal, lo que equivale a una evaluación de 0 a 10. Estos perfiles no son mutuamente excluyentes, por lo que se pueden obtener valores máximos (1.0) en múltiples o incluso en todos los perfiles.

A la vista de la situación expuesta, este trabajo de fin de grado pretende dar una solución a la situación planteada. Para ello se propondrán varios modelos de *deep learning* (DL) basados en dos arquitecturas diferentes, como son *BERT* y *LSTM*, posteriormente se aplicaran diferentes técnicas de aumento de datos que nos permitan mejorar el rendimiento de estos modelos. Tanto los modelos mencionados, como las técnicas de aumento de datos han dado resultados muy prometedores en el procesamiento del lenguaje natural. El mundo corporativo es un área donde los modelos basados en DL pueden generar

información valiosa. Por ello, se busca analizar las biografías de los directores independientes para llevar a cabo una modelización cualitativa automatizada de su perfil, con el objetivo de obtener una visión general de los consejos de administración de las empresas cotizadas en bolsa.

1.2. Objetivos

Hemos visto como las técnicas de *deep learning* enfocadas al procesamiento de datos no estructurado a ido en aumento en los últimos años, especialmente en ámbitos como el financiero o el corporativo. Este trabajo busca la creación e implementación diferentes modelos para el análisis de biografías de los consejeros independientes para poder obtener información relevante sobre los consejos de administración de las empresas que cotizan en el IBEX 35. Para ello se establecen dos objetivos principales.

- Creación de modelos de *deep learning* que permitan la clasificación automatizada de perfiles/subperfiles profesionales de los consejeros independientes.
- Implementación de técnicas de aumento de datos para mejorar los modelos iniciales, y así medir su impacto en la clasificación de los consejeros independientes.

Capítulo 2

Marco teórico

Durante este capítulo profundizaremos en los conceptos teóricos necesarios para comprender el funcionamiento desde lo más simple, una red neuronal básica, a lo que llevaremos a cabo en este trabajo como son transformadores y redes recurrentes.

2.1. Perceptrones

El perceptrón es un tipo de neurona artificial desarrollado por Frank Rosenblatt en los años 50 [27], diseñado como una aproximación simplificada a la neurona biológica. Se destaca en la historia de la inteligencia artificial por introducir conceptos fundamentales para el desarrollo de redes neuronales más complejas.

La función del perceptrón se basa en recibir múltiples entradas binarias, cada una multiplicada por un peso determinado, y generar una salida binaria. Esta salida se calcula como sigue:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{umbral} \\ 1 & \text{if } \sum_j w_j x_j > \text{umbral} \end{cases}$$

El concepto de umbral puede ser reemplazado por el de sesgo (b), simplificando la notación y el cálculo. La regla de decisión del perceptrón se puede reescribir de la siguiente forma:

$$\text{output} = \begin{cases} 0 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

El perceptrón puede modelar operaciones lógicas básicas como AND, OR, y NAND, mostrando su capacidad para realizar tareas de computación elemental. Sin embargo, es incapaz de resolver la función XOR, destacando una limitación importante de este modelo en problemas que no son linealmente separables.

El perceptrón jugó un papel crucial en el desarrollo inicial de la inteligencia artificial, influenciando el diseño de algoritmos de aprendizaje y la creación de modelos neuronales más avanzados. El entendimiento del perceptrón es vital para comprender los fundamentos teóricos de las redes neuronales modernas.

2.2. Funciones de activación

Una red neuronal sin una función de activación es esencialmente un modelo de regresión lineal clásico. La función de activación realiza la transformación no lineal de la entrada, lo que la hace capaz de aprender y realizar tareas más complejas.

En la entrada, una neurona artificial calcula una suma ponderada de su entrada y agrega un sesgo. Definimos $z = \sum_i x_i w_i + b$, $z \in \mathbb{R}$. Una vez calculado el valor z, la neurona artificial decide si se debe *activar* o no, basándose en el valor de z.

2.2.1. Función step

La función step es aquella basada en umbrales. Si el valor está por encima de cierto valor, se activa. Si es menor que el umbral, lo contrario, no se activa. El umbral más común es 0. La función de activación step se define como:

$$\text{STEP}(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

2.2.2. Función sigmoidea

Este tipo de funciones permiten mitigar el efecto de outliers en el entrenamiento de nuestro modelo. Esta función está contenida en el intervalo [0,1]. Además debido a su naturaleza, los valores muy extremos (muy positivos o muy negativos), producirán valores muy cercanos a los límites del intervalo. Estos valores pueden interpretarse como probabilidades.

$$sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

2.2.3. Función tangente hiperbólica

La función de activación tangente hiperbólica es similar a la función sigmoidea, aunque ésta toma valores en un rango [-1,1]. Por lo general funciona mejor que la función sigmoidea. Esta función suele usarse en capas ocultas de la red, ya que la media de sus valores es 0 o muy próximo a el, lo que facilita en el centrado de datos y el aprendizaje en la siguiente capa. Como podemos ver es una versión matemáticamente modificada de la función sigmoidea.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.2}$$

2.2.4. Función softmax

La función softmax es esencial en las redes neuronales, especialmente en las tareas de clasificación multinomial, donde cada entrada puede pertenecer a una o varias clases. Esta función actúa como una generalización de la regresión logística y se adapta bien tanto a datos continuos como categóricos. Se utiliza típicamente en la capa de salida de clasificadores para convertir los valores de entrada en una distribución de probabilidad.

Cada valor de salida representa la probabilidad de que la entrada pertenezca a una clase específica entre todas las clases posibles, con la característica de que la suma de todas estas probabilidades es igual a uno. Este comportamiento asegura que las salidas se pueden interpretar directamente como probabilidades.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.3}$$

2.3. Entrenamiento

El entrenamiento de una red neuronal implica varios conceptos clave: funciones de pérdida, descenso del gradiente, optimizadores y retropropagación. Estos conceptos son clave para entender el funcionamiento básico de una red neuronal.

2.3.1. Funciones de pérdida

La función de pérdida, también conocida como función de costo, mide la discrepancia entre las predicciones realizadas por el modelo y los valores reales del conjunto de entrenamiento. El objetivo es minimizar esta función a través del entrenamiento. Algunas de las funciones de pérdida más conocidas son:

Error cuadrático medio (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Donde y_i son los valores verdaderos, $\hat{y}_i$ son las predicciones del modelo, y n es el número de ejemplos. El MSE penaliza más los errores grandes debido al cuadrado de la diferencia.

Entropía cruzada

La entropía cruzada se utiliza comúnmente en problemas de clasificación, especialmente en clasificación binaria y multiclase. Para clasificación binaria, la fórmula es:

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Para clasificación multiclase, se usa la versión generalizada:

$$L = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

donde C es el número de clases, $y_{i,c}$ es un indicador binario que es 1 si la clase verdadera es c y 0 en caso contrario, y $\hat{y}_{i,c}$ es la probabilidad predicha de la clase c para el ejemplo i.

2.3.2. Retropropagación

La retropropagación es el algoritmo estándar para entrenar redes neuronales, permitiendo calcular los gradientes de la función de pérdida con respecto a cada peso mediante la regla de la cadena. Este proceso se divide en dos fases:

Propagación hacia adelante

En esta fase, se pasan los datos de entrada a través de la red para obtener las predicciones. Para una red con capas l, $l+1$, ..., la salida de una capa se calcula como:

$$a^{(l+1)} = f(z^{(l+1)})$$
$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$

donde $a^{(l)}$ son las activaciones de la capa l, $W^{(l)}$ son los pesos, $b^{(l)}$ son los sesgos, y f es la función de activación.

Propagación hacia atrás

En esta fase, se calculan los gradientes de la función de pérdida con respecto a cada peso usando la regla de la cadena. Para la última capa, el gradiente de la pérdida con respecto a las activaciones es:

$$\delta^{(L)} = \nabla_a L \circ f'(z^{(L)})$$

donde $\circ$ denota la multiplicación término a término y f' es la derivada de la función de activación.

Para capas anteriores, el gradiente se propaga hacia atrás como:

$$\delta^{(l)} = ((W^{(l+1)})^T \delta^{(l+1)}) \circ f'(z^{(l)})$$

Finalmente, los gradientes de la función de pérdida con respecto a los pesos y sesgos son:

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l+1)}(a^{(l)})^T$$
$$\frac{\partial L}{\partial b^{(l)}} = \delta^{(l+1)}$$

Este proceso iterativo permite ajustar los pesos de la red para minimizar la función de pérdida y mejorar la precisión de las predicciones.

2.3.3. Descenso del gradiente

El descenso del gradiente es un algoritmo para minimizar la función de pérdida ajustando los pesos de la red [23]. La idea es actualizar los pesos en la dirección opuesta al gradiente de la función de pérdida con respecto a los pesos. La actualización de los pesos w se realiza de la siguiente manera:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

donde η es la tasa de aprendizaje y $\frac{\partial L}{\partial w}$ es el gradiente de la pérdida con respecto a los pesos. A continuación se muestra la figura donde se representan las etapas del descenso del gradiente.

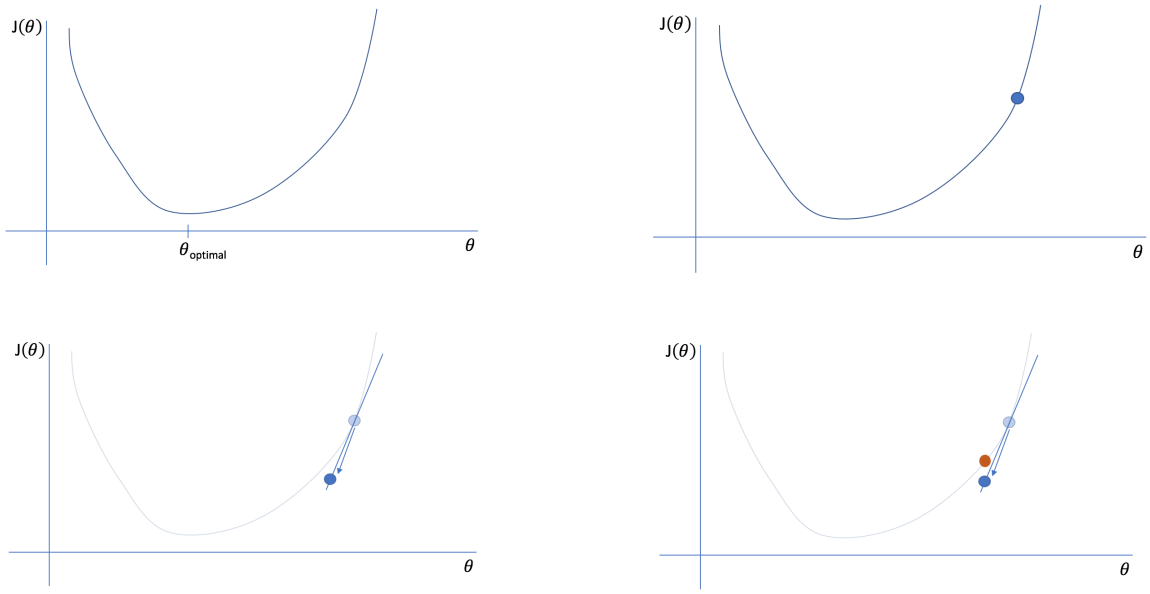


Figura 2.1: Pasos del cálculo del descenso del gradiente

2.3.4. Optimizadores

Los métodos de optimización son variantes del descenso del gradiente que mejoran la eficiencia y convergencia del entrenamiento.

Momentum

El método de momentum acelera el SGD (Stochastic Gradient Descent) acumulando un término de "velocidad"que suaviza las actualizaciones:

$$v_t = \gamma v_{t-1} + \eta \nabla L(w_t)$$

$$w_{t+1} = w_t - v_t$$

donde γ es el coeficiente de momentum.

RMSprop

RMSprop ajusta la tasa de aprendizaje para cada peso basándose en la media cuadrática de los gradientes:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

donde ρ es un parámetro de decaimiento y ϵ es un pequeño término para evitar divisiones por cero.

Adam (Adaptive Moment Estimation)

Adam combina las ideas de momentum y RMSprop, manteniendo un promedio móvil de los gradientes y sus cuadrados:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

donde m_t y v_t son estimaciones del primer y segundo momento del gradiente respectivamente. Las actualizaciones de los pesos son:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Adam es popular debido a su robustez y capacidad para manejar problemas con gradientes ruidosos y dispersos.

En nuestro caso específico, para el abordaje de problemas en procesamiento del lenguaje natural (PLN), el optimizador Adam (Adaptive Moment Estimation) suele ser el más usado para manejar gradientes ruidosos y dispersos, características comunes en PLN. Adam combina las ventajas del descenso de gradiente estocástico con momentum y RMSprop, ajustando las tasas de aprendizaje de manera adaptativa para cada parámetro, lo que resulta en una convergencia más rápida y estable. Además, su robustez lo hace adecuado para entrenar modelos complejos como redes neuronales recurrentes (RNNs) y transformadores, comúnmente utilizados en PLN.

2.4. Reguralización del entrenamiento

Como ya sabemos, las redes neuronales tienen la capacidad de aprender patrones complejos en los datos de entrenamiento. Tras el entrenamiento de nuestro modelo cabe la posibilidad que el modelo no se ajuste correctamente o por el contrario que lo haga de una forma excepcional, es decir, que el modelo se vuelve demasiado especializado en los datos específicos de entrenamiento y pierde la capacidad de generalizar a datos nuevos. Es lo que se conoce como *underfitting* y *overfitting*.

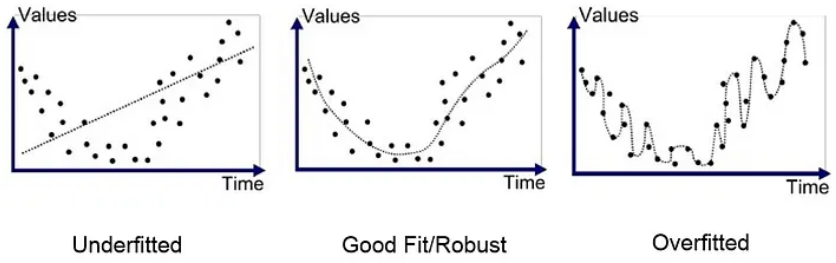


Figura 2.2: Ejemplo de infrajauste, ajuste óptimo y sobreajuste

La regularización es la técnica que nos permite controlar el sobreajuste en las redes neuronales. Al aplicar técnicas de regularización, se controla la complejidad del modelo y se evita que los pesos y parámetros se vuelvan demasiado grandes o especializados en los datos de entrenamiento. A continuación se muestran algunas de las técnicas de regularización más comunes:

- Regularización L1 y L2: se basan en agregar términos de penalización a la función de pérdida para desincentivar modelos complejos. Por un lado la regularización L1 se basa en añadir una penalización igual al valor absoluto de la magnitud de los coeficientes, es decir, limita el tamaño de los coeficientes. Se expresa:

$$\text{L1 Regularization:} \quad \lambda \sum_{j=1}^{p} |\beta_j| \tag{2.4}$$

  La regularización L2 añade una penalización igual al cuadrado de la magnitud de los coeficientes. L2 no producirá modelos dispersos y todos los coeficientes se reducen por el mismo factor. Se expresa:

$$\text{L2 Regularization:} \quad \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2.5}$$

  A continuación se puede ver una representación de ambas regularizaciones.

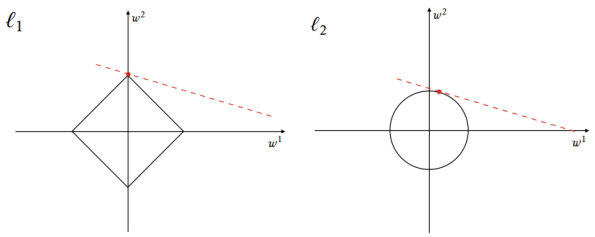


Figura 2.3: Comparativa representación L1 y L2 [11]

- Parada Temprana (Early Stopping) [21]: Es una técnica que implica interrumpir el proceso de entrenamiento cuando el rendimiento del modelo en el conjunto de datos de validación comienza a reducirse.

- Dropout: Técnica de regularización que apaga aleatoriamente un porcentaje de neuronas durante el entrenamiento de la red neuronal. Esta estrategia evita que el modelo se vuelva dependiente de un conjunto específico de características al hacer que cada neurona tenga que aprender de manera más independiente. De esta forma se previene el sobreajuste y se promueve una mejor generalización del modelo.

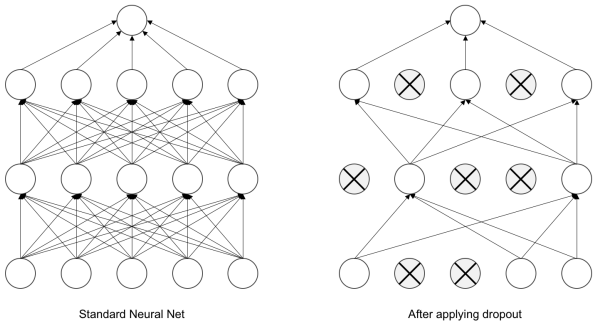


Figura 2.4: Ejemplo aplicación técnica de regularización dropout

2.5. Aumento de datos

En el ámbito del procesamiento del lenguaje natural, mientras que para la creación de modelos de lenguaje se pueden emplear conjuntos de datos extensos como Wikipedia o libros, la situación se vuelve más desafiante al abordar tareas específicas como la clasificación de reseñas de películas o la detección de spam, donde la disponibilidad de conjuntos de datos etiquetados es limitada. En estos casos, el aumento de datos se convierte en una estrategia crucial. Se refiere a técnicas que buscan incrementar la diversidad en el conjunto de entrenamiento sin necesidad de recolectar nuevos datos explícitamente. A diferencia del procesamiento de imágenes, aplicar estas técnicas al NLP representa un desafío adicional debido a la naturaleza discreta del lenguaje y sus reglas gramaticales, que introducen ruido y dificultan la generación de modelos robustos.

Un aumento de datos efectivo debe ser fácil de implementar y mejorar el rendimiento del modelo entrenado. Sin embargo, los métodos actuales suelen requerir un compromiso entre la simplicidad y el rendimiento. Dependiendo del modelo y del contexto específico, es necesario buscar una técnica que logre mantener un equilibrio adecuado entre ambos aspectos. Además, es una herramienta crucial en situaciones donde los datos etiquetados son limitados, facilitando el aprendizaje sin necesidad de grandes volúmenes de datos anotados manualmente. A continuación, exploraremos algunas de estas técnicas.

Cabe destacar que en el procesamiento del lenguaje natural, en particular en el idioma español, hay una falta de técnicas bien definidas que permitan incrementar un conjunto de datos.

Vamos a dividir las técnicas en dos grandes grupos que denominaremos Aumento Simbólico (Symbolic augmentation) y Aumento Neuronal (Neural Augmentations). Su diferencia radica en la utilización de redes auxiliares u otro tipos de modelos estadísticos para la generación de nuevos datos. A continuación se explicaran las tres técnicas más

comunes de Aumento Simbólico.

2.5.1. Aumento simbólico

Vamos a diferenciar 3 técnicas diferentes (Aumento basado en reglas, aumento del espacio de características y aumento mixUp).

Aumento basado en reglas

El aumento basado en reglas se basa en la utilización de reglas predefinidas para modificar y generar nuevos ejemplos de datos [10]. Podemos dividir estas reglas en 4:

- Reemplazo de Sinónimos (SR): Elige al azar n palabras de la oración (no palabras vacías). Cada una de ellas es reemplazada por un sinónimo elegido al azar.
- Inserción Aleatoria (RI): Encuentra un sinónimo aleatorio de una palabra aleatoria en la oración (no palabra vacía). Se inserta ese sinónimo en una posición aleatoria de la oración. Posteriormente se repite el proceso n veces.
- Intercambio Aleatorio (RS): Elige al azar dos palabras en la oración e intercambia sus posiciones. Repitiendo este proceso n veces.
- Eliminación Aleatoria (RD): Elimina aleatoriamente cada palabra en la oración con una probabilidad p.

Aumento en espacio de características

Esta técnica implica agregar ruido o modificar las representaciones intermedias en el espacio de características de las redes neuronales. Una técnica muy conocida en esta categoría es el *label smoothing*.

La técnica de *label smoothing* introduce ruido en las etiquetas de los datos para considerar posibles errores en los conjuntos de datos, lo que ayuda a evitar que los modelos de aprendizaje automático se vuelvan excesivamente confiados en sus predicciones. Al ajustar ligeramente las etiquetas, de valores estrictos de 0 y 1 a valores suavizados como 0,933 y 0,033 (suponiendo un ϵ de 0.1), se crea un modelo más robusto frente a errores y con mejor capacidad de generalización.

Este enfoque actúa como una forma de regularización, similar al *dropout*, previniendo el sobreajuste del modelo a los datos de entrenamiento. Al suavizar las etiquetas, *label smoothing* ayuda a manejar mejor los errores en el conjunto de datos y reduce la confianza excesiva del modelo en sus predicciones, mejorando así su rendimiento en datos no vistos.

MixUp

Mixup es una las técnicas más recientes en el campo del aumento de datos. Se basa en la interpolación lineal de las entrada con sus etiquetas correspondientes. Ha demostrado una gran eficacia en la clasificación de imágenes al interpolarlas a nivel de píxel.

Actualmente hay varios estudios que aplican esta técnica al campo del procesamiento del lenguaje con éxito [33].

2.6. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) o (NLP por sus siglas en inglés) es una rama de la inteligencia artificial que se enfoca en la interacción entre computadoras y humanos a través del lenguaje natural. La historia del PLN comienza en 1950, cuando Alan Turing publicó *"Computing Machinery and Intelligence"*, donde propuso el test de Turing como un criterio cualitativo para medir la inteligencia de las máquinas[37].

El procesamiento del texto natural presenta diversas dificultades debido a la ambigüedad inherente del lenguaje en varios niveles:

- Nivel Léxico: Una misma palabra puede tener múltiples significados.
- Nivel Referencial: Las palabras pueden referirse a diferentes cosas dependiendo del contexto.
- Nivel Estructural: La estructura gramatical puede variar y afectar el significado.
- Nivel Pragmático: Las oraciones pueden no significar literalmente lo que dicen, esto depende del contexto.

2.6.1. Tokenización

En NLP el proceso de convertir nuestras secuencias de caracteres, palabras o párrafos en inputs para la computadora se llama tokenización. Se puede pensar en el token como la unidad para procesamiento semántico.

La tokenización es el primer paso en cualquier proceso de PLN (procesamiento del lenguaje natural). Tiene un efecto importante en el resto del proceso. Un tokenizador divide datos no estructurados y texto en lenguaje natural, en fragmentos de información que pueden considerarse como elementos discretos.

Encontramos diferentes tipos de tokenizadores. Todos con el mismo objetivo, obtener valores que sean cortos pero muy significativos para el modelo. A continuación se enumeran algunos de los tokenizadores [20].

- Tokenización Word-based: divide un fragmento de texto en palabras basándose en un delimitador. El delimitador más comúnmente utilizado es el espacio. También puedes dividir tu texto utilizando más de un delimitador, como el espacio y los signos de puntuación. Dependiendo del delimitador que utilices, obtendrás diferentes tokens a nivel de palabras.

  "Is it weird I don't like coffee?"
  ["Is", "it", "weird", "I", "don't", "like", "coffee?"]

- Tokenización *Character-based*: este tipo de tokenizador separa el texto en caracteres, y no en palabras. Esto nos proporciona un vocabulario mucho más corto y habrá muchos menos tokens por fuera del vocabulario conocido.

- Tokenización por *Subword*: Los algoritmos de tokenización de subpalabras se basan en el principio de que las palabras de uso frecuente no deben dividirse, mientras que las palabras raras deben descomponerse en subpalabras significativas.

2.6.2. Numeralización

La numeralización es el proceso de mapear los tokens generados en la tokenización a números. El proceso de la numeralización se puede resumir en dos sencillos pasos:

1. Creación de vocabulario. Para ello se hace una lista con todas las posibles palabras que puede tener tu entrada.

2. Reemplazar cada palabra por el índice en del vocabulario creado.

A continuación se muestra un ejemplo:

Texto Tokenizado: xxbos xxmaj this movie , which i just xxunk at the video store , has apparently sit around for a'

Numeralizado: [2, 8, 21, 28, 11, 90, 18, 59, 0, 45, 9, 351, 499, 11, 72, 533, 584, 146, 29, 12]

2.6.3. División en lotes

Para entrenar un modelo de lenguaje, es necesario dividir el texto en lotes (batches). A diferencia de las imágenes, no se puede redimensionar el texto a una longitud fija. En cambio, se divide el texto en secuencias de longitud fija y se agrupan en lotes de manera que cada nuevo lote comience donde terminó el anterior, preservando el orden. A continuación se muestra un ejemplo de división por lotes:

Texto tokenizado:

: xxbos xxmaj en este capítulo , revisaremos el ejemplo de clasificar reseñas de películas que estudiamos en el capítulo 1 y profundizaremos bajo la superficie . xxmaj primero veremos los pasos de procesamiento necesarios para convertir el texto en números y cómo personalizarlo . xxmaj haciendo esto , tendremos otro ejemplo del preprocesador utilizado en la API de bloques de datos . xxmaj luego estudiaremos cómo construir un modelo de lenguaje y entrenarlo durante un tiempo.

División en lotes:

Para un tamaño de lote de 6, el texto se divide en partes contiguas de longitud 15:

Parte 1: xxbos xxmaj en este capítulo , revisaremos el ejemplo de clasificar
Parte 2: reseñas de películas que estudiamos en el capítulo 1 y
Parte 3: profundizaremos bajo la superficie . xxmaj primero veremos
Parte 4: los pasos de procesamiento necesarios para convertir el texto en
Parte 5: números y cómo personalizarlo . xxmaj haciendo esto ,

Parte 6: tendremos otro ejemplo del preprocesador utilizado en la API de

El tamaño de los lotes o *batch size* es uno de los hiperparámetros más importantes en una red neuronal. Un tamaño de lote grande significa más consumo de memoria y mayor rapidez de entrenamiento de la red, ya que se procesa una gran cantidad de datos en paralelo. Si elegimos un *batch size* pequeño, el entrenamiento será más lento, se consumirá menos memoria y la calidad de ajuste será mejor [26].

2.6.4. Modelo de lenguaje

Una vez se dispone de los datos de entrenamiento se define lo que conocemos como modelo de lenguaje. Un modelo de lenguaje es un tipo de modelo de inteligencia artificial diseñado para predecir la probabilidad de una secuencia de palabras. Estos modelos, tratan de comprender y generar texto que se parezca lo máximo posible al lenguaje humano natural. Estos modelos son entrenados en grandes corpus de texto en diferentes idiomas para aprender sus patrones y estructuras.

Como se ha mencionado, el objetivo principal de un modelo de lenguaje es predecir la próxima palabra en una secuencia, dada la secuencia anterior. La probabilidad condicional de una palabra w_m dado un contexto $w_1 w_2 \dots w_{m-1}$ se puede calcular como:

$$P(w_m \mid w_1 w_2 \dots w_{m-1}) = \frac{C(w_1 w_2 \dots w_m)}{\sum_{w \in V} C(w_1 w_2 \dots w_{m-1} w)}$$

donde V es el vocabulario del modelo.

Vamos a diferenciar entre dos tipos de modelos de lenguaje:

- N-gramas: Estos modelos consideran una secuencia de n palabras y predicen la siguiente palabra basándose únicamente en las n-1 palabras anteriores. Aunque son simples y eficientes, tienen limitaciones a la hora de capturar dependencias a largo plazo.
- Modelos Basados en Redes Neuronales: Estos modelos utilizan redes neuronales recurrentes (*RNN*), redes neuronales convolucionales (*CNN*) y transformadores (*GPT* y *BERT*). Estos modelos son capaces de capturar dependencias a largo plazo y manejar contextos más complejos en el lenguaje.

Durante el desarrollo de este trabajo fin de grado nos enfocaremos en el segundo tipo de modelos de lenguaje. Más concretamente en el uso de *LSTMs*, un tipo concreto de RNN y un modelo de Transformadores.

2.7. Redes neuronales recurrentes

Las redes neuronales recurrentes (RNNs siglas en Ingles) son un tipo de arquitectura de red neuronal cuyo uso principal es la detección de patrones en datos secuenciales, ya sea texto, voz, series temporales [31]. Si nos centramos en las aplicaciones dentro del lenguaje estas redes tienen aplicaciones para la generación de texto y reconocimiento de voz.

Lo que diferencia las Redes Neuronales Recurrentes de las *MLPs (Feed Forward Neural Networks)*, mencionado anteriormente, es como la información pasa a través de de la red. Mientras que las *Feedforward Networks* nos permiten el paso de información por la red sin ciclos, es decir, en una única dirección, las RNN permiten conexiones cíclicas, o lo que es lo mismo, el paso de información de vuelta a ellas mismas. En la figura 2.5 se puede ver la diferencia. Esto nos permite tener una memoria interna de los secuencias anteriores (X0_t-1) y no solo de las secuencias actuales Xt. Estas diferencias se pueden apreciar en la figura 2.6. La figura muestra un desglose de la capa oculta de la RNN para ver su funcionamiento en detalle.

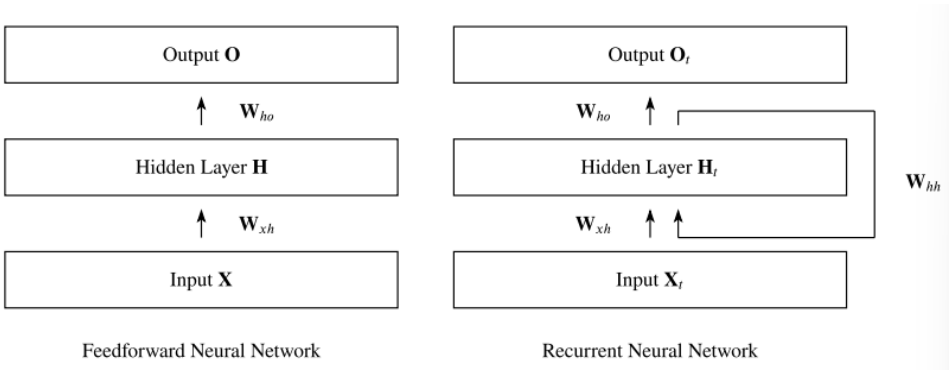


Figura 2.5: Capa oculta de una RNN

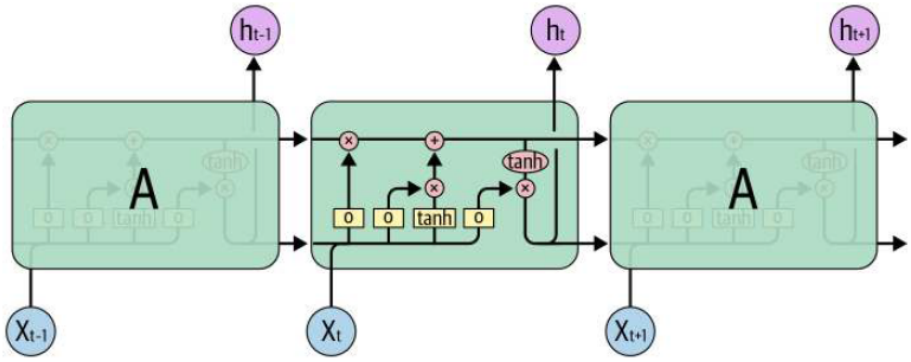


Figura 2.6: Desglose Capa oculta de una RNN

Las conexiones que forman ciclos dentro de la red permite que la información de las entradas anteriores influya en las salidas actuales. Esto se representa matemáticamente como:

$$H_t = \phi_h(X_t W_{xh} + H_{t-1} W_{hh} + b_h)$$

Donde

- H_t es el estado oculto en el tiempo t.
- X_t es la entrada en el tiempo t.
- W_{xh} y W_{hh} son matrices de pesos.

- b_h es el sesgo.

Vemos ahora cómo propagamos el error para RNNs. Hemos visto que para las MLPs utilizamos la técnica de *backpropagation*. Para las redes recurrentes no se usa la misma técnica, la técnica se conoce como *Backpropagation Through Time* (BPTT).

Esta técnica es una adaptación de la técnica de *backpropagation* para RNNs. En BPTT, la red se *desenrolla* a través del tiempo, creando una red de alimentación directa equivalente para cada paso temporal. La pérdida total se calcula como la suma de las pérdidas en cada paso temporal. Esta pérdida se expresa de la siguiente forma:

$$L(O, Y) = \sum_{t=1}^{T} \ell_t(O_t, Y_t)$$

Donde

- ℓ_t es la función de pérdida en el tiempo t. ℓ_t puede ser diferente dependiendo del problema planteado (MSE, Cross Entropy, etc.).
- O_t es la salida en el tiempo t.
- Y_t es el valor objetivo en el tiempo t.

Observamos que después de aplicar la regla de la cadena para obtener las derivadas parciales surge un problema. La función de pérdida puede ser extremadamente larga. Para abordar este problema se utiliza una variante denominada *Truncated BPTT*, donde se establece un límite superior en el número de pasos temporales, en el que el gradiente puede retroceder, es decir, se define una ventana de tiempo móvil que limita la cantidad de estados anteriores considerados durante la actualización de los pesos.

Uno de los problemas más significativos al entrenar RNNs es el gradiente desaparecido y el gradiente explosivo:

- **Gradiente Desaparecido**: Ocurre cuando los valores en la matriz de pesos son pequeños, haciendo que los gradientes disminuyan exponencialmente a medida que retroceden en el tiempo. Esto impide que la red aprenda de entradas lejanas en la secuencia.
- **Gradiente Explosivo**: Ocurre cuando los valores en la matriz de pesos son grandes, haciendo que los gradientes crezcan exponencialmente, lo que puede resultar en actualizaciones muy grandes de los pesos, desestabilizando el entrenamiento.

Estos problemas motivaron el desarrollo de unidades más avanzadas como las *Long Short-Term Memory* (LSTM), que están diseñadas específicamente para mitigar el problema del gradiente desaparecido al mantener un gradiente constante durante muchas iteraciones.

2.7.1. LSTMS

Uno de los atractivos de las RNNs es la idea de que podrían conectar información previa con la tarea actual. Si las RNNs pudieran hacer esto, serían extremadamente útiles. Pero, ¿pueden hacerlo? Depende.

A veces, solo necesitamos mirar la información reciente para realizar la tarea actual. Por ejemplo, considera un modelo de lenguaje que intenta predecir la siguiente palabra basándose en las anteriores. Si estamos tratando de predecir la última palabra en *las nubes están en el cielo*, no necesitamos más contexto: es bastante obvio que la siguiente palabra será *cielo*. En tales casos, donde la brecha entre la información relevante y el lugar donde se necesita es pequeña, las RNNs pueden aprender a usar la información pasada. Pero hay casos donde se necesita más contexto. Pongamos como ejemplo la frase "He crecido en Francia en un pequeño pueblo ... por lo que hablo Francés". La información reciente indica que la siguiente palabra debe ser el nombre del idioma pero necesitamos el contexto del país, Francia, del inicio de la frase. Desgraciadamente cuando el hueco crece, las RNN son incapaces de conectar correctamente la información. Por suerte disponemos de las LSTMs que carecen de este problema.

Las redes Long Short Term Memory, más conocidas como "LSTMs" son un tipo especial de redes capaces de aprender dependencias a largo plazo. Fueron introducidas por primera vez en 1997 por Hochreiter y Schmidhuber [32]. Están diseñadas principalmente para evitar el problema de las largas dependencias.

A continuación se muestra el paso a paso de una LSTM.

- El primer paso en una LSTM consiste en determinar qué información se descartará del estado de la célula. Esta decisión se toma mediante una capa sigmoidea denominada *forget gate layer*. Esta capa examina ht-1 y xt, y genera un valor entre 0 y 1 para cada número en el estado de la célula Ct-1. Un valor de 1 representa conservar completamente esta información, mientras que un valor de 0 representa descartar completamente esta información.

- Decisión sobre la nueva información a almacenar: El siguiente paso es decidir qué nueva información se almacenará en el estado de la célula, proceso que consta de dos partes. Primero, una capa sigmoidea llamada *capa de la puerta de entrada* decide qué valores se actualizarán. A continuación, una capa tanh crea un vector de nuevos valores candidatos, C t, que podrían añadirse al estado. En el paso siguiente, estos dos elementos se combinarán para actualizar el estado.

- Actualización del estado de la célula: En esta etapa, se actualiza el antiguo estado de la célula, Ct-1, al nuevo estado de la célula, Ct. Los pasos previos ya han determinado qué hacer, por lo que solo es necesario ejecutarlo. Se multiplica el estado antiguo por ft, descartando la información que se decidió olvidar previamente. Luego, se añade C t, que representa los nuevos valores candidatos, escalados según el grado en que se decidió actualizar cada valor del estado.

- Decisión sobre la salida a generar: Finalmente, se debe decidir qué información se va a emitir. Esta salida se basará en el estado de la célula, pero será una versión filtrada del mismo. Primero, se utiliza una capa sigmoidea que decide qué partes

del estado de la célula se van a emitir. Luego, el estado de la célula pasa por una función tanh (para limitar los valores entre -1 y 1) y se multiplica por la salida de la puerta de salida, de modo que solo se emitan las partes que se han decidido previamente.

2.8. Transformers

Los *transformers* son la última incorporación en cuanto a modelos de *deep learning* para PLN. Estos modelos se han puesto a la cabeza en casi todas las tareas principales de PLN en los últimos dos años. Modelan el contexto textual, pero no de manera secuencial. Dada una palabra en la entrada, prefieren mirar todas las palabras a su alrededor (conocido como autoatención) y representar cada palabra con respecto a su contexto. La autoatención es un mecanismo de atención que relaciona diferentes posiciones de una sola secuencia para calcular una representación de la secuencia. Los *transformers* pueden modelar dicho contexto y, por lo tanto, se han utilizado ampliamente en tareas de PLN debido a esta mayor capacidad de representación en comparación con otras redes profundas.

2.8.1. Arquitectura

La arquitectura de los *transformers* se basa en una estructura *encoder-decoder*. El codificador asigna una secuencia de entrada de representaciones de símbolos (x1, ..., xn) a una secuencia de representaciones continuas z = (z1, ..., zn). Posteriormente, el decodificador genera una secuencia de salida (y1, ..., ym) de símbolos, de uno en uno. En cada paso, el modelo es autorregresivo, es decir, consume los símbolos previamente generados como entrada adicional al generar el siguiente.

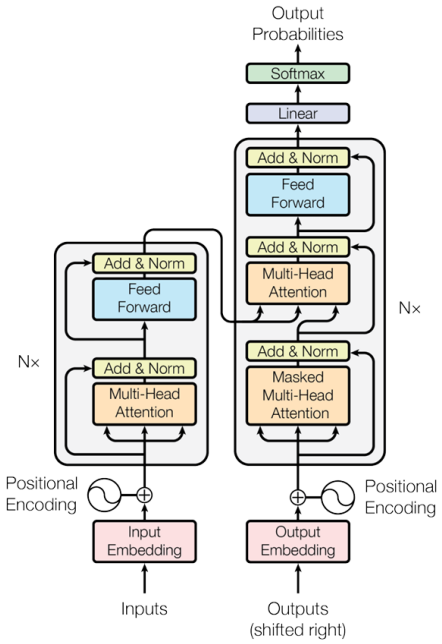


Figura 2.7: Arquitectura Transformador

El *transformer* sigue la arquitectura general de la figura 2.7. utilizando capas apiladas

de auto-atención y capas totalmente conectadas punto a punto tanto para el codificador como para el decodificador. El codificador se encuentra a la izquierda de la figura mientras que el decodificador se encuentra a la derecha [36].

- Encoder: El encoder está compuesto por una pila de N = 6 capas idénticas. Cada capa tiene dos subcapas. La primera es un mecanismo de autoatención multi-cabezal 2.8, y la segunda es una red neuronal totalmente conectada y posicionada. Empleamos una conexión residual alrededor de cada una de las dos subcapas, seguida de capa de normalización. Es decir, la salida de cada subcapa es LayerNorm(x + Sublayer(x)), donde Sublayer(x) es la función implementada por la subcapa en si misma. Para facilitar estas conexiones residuales, todas las subcapas en el modelo, así como las capas de *embedding*, producen salidas de dimensión d_model = 512.
- Decoder: El decoder también está compuesto por una pila de N = 6 capas idénticas. Como se observa en la figura 2.7 tiene las mismas subcapas que el codificador. A mayores se le añade una capa de atención enmascarada con el objetivo de evitar entradas anteriores se fijen en entradas posteriores.

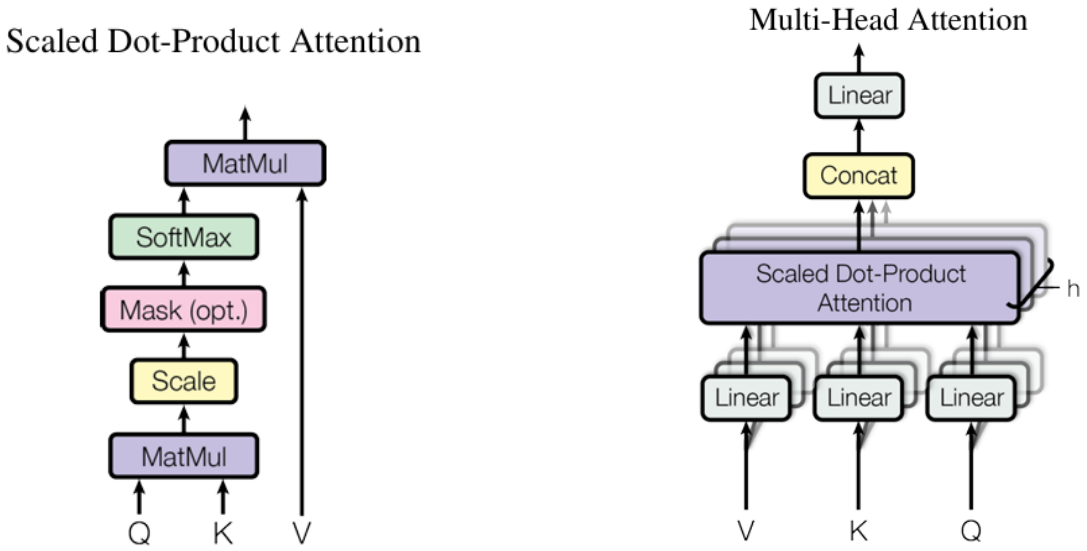


Figura 2.8: Interior de una capa de atención (izquierda) y capa de atención en paralelo

La función de atención se puede describir como un mapeo de una consulta y un conjunto de pares clave-valor a una salida, donde la consulta, las claves, los valores y la salida son todos vectores. La salida se calcula como una suma ponderada de los valores, donde el peso asignado a cada valor se calcula mediante una función de compatibilidad de la consulta con la clave correspondiente. Pongamos un ejemplo para una mejor comprensión. Supongamos que disponemos de sistema de búsquedas de una biblioteca. La *"query"* es lo que el usuario busca , las "keys"son las descripciones de los libros, y los *"values"* son los propios libros. La función de atención calcula la salida como una suma ponderada de los valores, donde los pesos asignados a cada valor se calculan en función de la compatibilidad de la de la búsqueda realizada (*query*) con la descripción del libro (*key*), utilizando una función de similitud como el producto punto. Así, el libro más relevante para la consulta

tendrá el mayor peso y será el más destacado en la salida.

Para poder calcular la atención, se utilizan 3 matrices llamadas "Query-Key-Value" que van a operar siguiendo una fórmula que nos devuelve un "score" o puntuaje de atención.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.6}$$

En la matriz Q de Query tendremos los tokens que estamos evaluando. En la matriz K de Key tendremos los tokens nuevamente, como claves del diccionario. En la matriz V de Value tendremos todos los tokens "de salida".

Capítulo 3

Contexto

En este capítulo se explicaran las tecnologías utilizadas para el desarrollo del TFG, así como introducción sobre el Ibex35, sus Consejos de Administración y los consejeros independientes.

3.1. Contexto financiero

3.1.1. Empresas que cotizan en el Ibex35

El Ibex 35 (acrónimo de Iberian Index, Índice ibérico) se creó el 14 de enero de 1992 y está gestionado por Bolsas y Mercados Españoles (BME).

Es el índice bursátil de la Bolsa española y refleja la evolución de las 35 compañías más negociadas por los inversores de entre todas las que cotizan en el Sistema de Interconexión Bursátil Electrónico (SIBE) en las cuatro bolsas españolas (Madrid, Barcelona, Bilbao y Valencia).

El Ibex 35 está formado por aquellas compañías designadas por el comité asesor técnico que cada seis meses se reúne para decidir si debe o no producirse algún cambio sobre la composición de empresas del Ibex35.

Para formar parte del Ibex35 se deben de cumplir dos requisitos. Se mencionan a continuación [7]:

- **Volumen**: El Comité Asesor del Ibex computa el volumen de contratación en euros en el mercado de órdenes durante el periodo de control, los seis meses previos a la reunión. Además, considera diferentes factores para garantizar la calidad de dicho volumen. Entre otros, analiza aspectos como el número de operaciones contratadas o posibles cambios en el accionariado.
- **Capitalización Media**: Solo pueden formar parte del IBEX 35 aquellos valores cuya capitalización media sea superior al 0,30 % de la capitalización media del selectivo durante el mencionado periodo de control. En todo caso, las normas técnicas consideran el capital flotante de los diferentes valores y, para ello, establecen diferentes coeficientes a aplicar sobre la capitalización media de los mismos de tal forma que se penaliza a aquéllos cuyo capital flotante es menor.

3.1.2. Consejos de Administración

El Consejo de Administración es un órgano directivo que se encarga de gestionar la política de una empresa (sociedad anónima o limitada), y de supervisar la dirección ejecutiva de la misma.

La ley impone un mínimo de tres miembros y un máximo de doce para las sociedades limitadas. Pero en las sociedades anónimas (todas las empresas del Ibex) no hay límite máximo, aunque se aconseja un máximo de 15. Dentro de tal límite, los estatutos sociales pueden fijar un número concreto de miembros o el número mínimo y máximo, en cuyo caso corresponde a la Junta (esto es, a los socios) la determinación del número concreto de sus componentes en cada momento.

El buen funcionamiento de un Consejo de Administración depende, desde luego, de los miembros que lo conforman. Es decir, los consejeros y, en especial, el presidente que es quien dirige el Consejo.

Los consejeros son los miembros del Consejo de Administración y con ello los responsables finales de las decisiones acordadas. Estos miembros son seleccionados por la asamblea general de accionistas, conforme a lo establecido en los estatutos de la organización.

Existen varios tipos de consejeros diferentes [8], a continuación se nombran tres:

- **Ejecutivos**: Miembro del Consejo que, además de sus funciones como consejero, interviene en la actividad diaria de la compañía, al desempeñar tareas de alta dirección, o bien es empleado.
- **Dominical**: Aquél que forma parte del Consejo de Administración porque posee una participación accionarial igual o superior a la que se considere legalmente significativa o por su condición de accionista aunque su participación accionarial no alcance dicha cuantía. Puede formar parte del consejo de forma directa o a través de un representante.
- **Independientes**: Miembro del Consejo de Administración que debe desempeñar sus funciones sin verse condicionado por relación alguna con la sociedad, con el equipo gestor, ni con los accionistas de control. Se elige en atención a sus circunstancias personales y profesionales. Su misión fundamental es defender los intereses de todos los accionistas y, en particular, de los minoritarios que no tienen acceso a un puesto en el Consejo.

Por otro lado tenemos la figura del presidente del Consejo. El presidente es el encargado de dirigir el Consejo. Su misión es coordinar para que el Consejo de Administración funcione correctamente, por lo que ha de procurar un entorno abierto y participativo. Entre sus responsabilidades destaca: convocar y desarrollar el orden del día, así como presidir y coordinar las reuniones, para procurar el buen funcionamiento y la organización del consejo. A continuación se muestran algunas de las funciones del Consejo de Administración.

- Velar por la misión de la empresa: El Consejo tiene la responsabilidad de concretar la misión que ha definido la empresa mediante el cumplimiento de una serie de objetivos estratégicos asumidos por la dirección.
- Tomar decisiones: La toma de decisiones sobre presupuestos, inversiones importantes, o enajenación de activos, así como el establecimiento de prioridades de actuación, permiten al Consejo adquirir cierto control en lo estratégico y en lo financiero para procurar la sostenibilidad de la compañía.
- Nombrar y destituir a la alta dirección: La empresa ha de contar con profesionales responsables en los cargos idóneos, es por ello que la designación del CEO le compete principalmente al Consejo.
- Orientar la estrategia: La manera en la que se ejecutarán los objetivos trazados, también define el tipo de empresa que se desea lograr.

3.1.3. Consejeros independientes

Como ya hemos mencionado en la sección anterior los consejeros independientes son un tipo de consejero dentro de los Consejos de Administración de las empresas. Son los encargados de velar por la correcta gestión, pero de manera externa. Su misión es aportar una visión independiente con el propósito de generar valor para los accionistas.

El fundamento de este TFG recae sobre este tipos de consejeros independientes por lo que es fundamental conocer su papel más a fondo dentro de los Consejos de Administración.

La elección de un consejero independiente se basa en sus cualidades personales y profesionales. Es crucial que estos individuos posean una experiencia sólida en el campo de actuación, un alto nivel de integridad y un historial de decisiones imparciales. Importante también, la ausencia de relaciones con la empresa, que su juicio sea objetivo y que sus decisiones no estén sesgadas por presiones internas o externas.

Dentro de las funciones del consejero independiente podemos destacar cuatro aspectos clave.

- Defensa de los intereses de los accionistas minoritarios: Estos accionistas, al no tener una representación significativa en el Consejo, pueden verse afectados por decisiones que beneficien únicamente a los accionistas mayoritarios. El objetivo del consejero independiente velar por estos accionistas para que las decisiones tomadas sean justas y equitativas para todos los accionistas.
- Supervisión y control del equipo directivo: La supervisión de las acciones del equipo directivo para asegurar que se alineen con los intereses de la empresa y sus accionistas.
- Evaluación de operaciones vinculadas y conflictos de interés: En diferentes situaciones como fusiones, adquisiciones pueden ocurrir conflictos de interés. Por esto la objetividad de los consejeros independientes les permite evaluar estas situaciones de manera imparcial, asegurando que las decisiones tomadas beneficien a la empresa en su conjunto.

- Mejora de la transparencia y la gobernanza: La presencia de consejeros independientes mejora la transparencia en la toma de decisiones. Al no tener vínculos con la empresa, pueden cuestionar decisiones y solicitar información adicional para asegurar que todas las decisiones se tomen con la mayor información y transparencia posible.

La inclusión de consejeros independientes en el Consejo de Administración es un pilar fundamental para una buena gobernanza corporativa. Su imparcialidad y objetividad mejoran la confianza de los inversores en la empresa, lo que puede traducirse en un mejor rendimiento financiero y una mayor valorización en el mercado.

3.2. Contexto tecnológico

3.2.1. Transfer learning & fine tuning

Transfer learning y *fine tuning* son términos muy similares pero con algunas diferencias. El *transfer learning* se da cuando usamos el conocimiento adquirido al resolver un problema específico y lo aplicamos a un problema nuevo pero que tiene relación. En la figura 3.1 se puede ver un visualización sobre diferentes ejemplos de *transfer learning*.

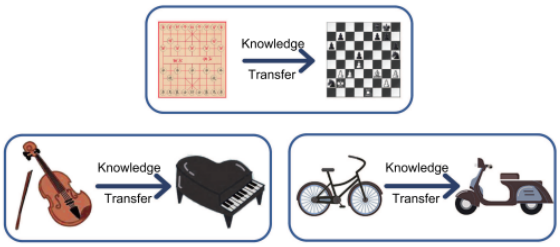


Figura 3.1: Ejemplo transfer learning

Por otro lado el *fine tuning* es la forma en la que se aplica el *transfer learning*. Concretamente, el *fine tuning* es un proceso que toma un modelo que ya ha sido entrenado para una tarea determinada, se vuelve a entrenar parcialmente el modelo preexistente en un conjunto de datos más pequeño y especializado, de modo que refine su comprensión para una tarea determinada. De esta forma es posible aprovechar las fortalezas generales de los modelos mientras se adaptan a necesidades particulares.

Supongamos que tenemos una nueva tarea similar a la original, utilizar una red neuronal artificial que ya ha sido diseñada y entrenada nos permite aprovechar lo que el modelo ya ha aprendido sin tener que desarrollarlo desde cero. Al construir un nuevo modelo desde 0 se deben de concretar muchos factores: número de capas, tipos de capas, orden de esas capas, número de nodos en cada capa, optimizador, tasa de aprendizaje, etc. Por lo cual si encontramos un modelo entrenado que ya haga bien una tarea, y esa tarea es similar a la nuestra, al menos, de alguna manera remota, entonces podemos aprovechar todo lo que el modelo ya ha aprendido y aplicarlo a nuestra tarea específica. Esto es lo que hace tan atractivo al *fine tuning*. Por otro lado, si las dos tareas son diferentes, entonces habrá cierta información que el modelo ha aprendido que puede no aplicarse a

nuestra nueva tarea. Para esto tenemos que hacer lo que se conoce como *afinado*.

Pongamos un ejemplo, supongamos que nosotros tenemos un modelo genérico BERT, y nuestro objetivo es la clasificación de reviews de Yelp. Para implementar el afinado se deberán eliminar las últimas capas de este modelo BERT y añadir capas necesarias para realizar la clasificación deseada. Esto se realiza ya que las capas al final de nuestro modelo pueden haber aprendido características que son muy específicas de la tarea original, mientras que las capas al comienzo del modelo generalmente aprenden características más generales. Además no queremos que los pesos de estas capas se actualicen cada vez que entrenemos el modelo con nuestros nuevos datos para nuestra nueva tarea. Queremos mantener todos estos pesos iguales a como estaban después de haber sido entrenados en la tarea original. Solo queremos que se actualicen los pesos en nuestras capas nuevas o modificadas. En la figura 3.2 se muestra una comparativa entre *fine tuning* y un modelo desde cero.

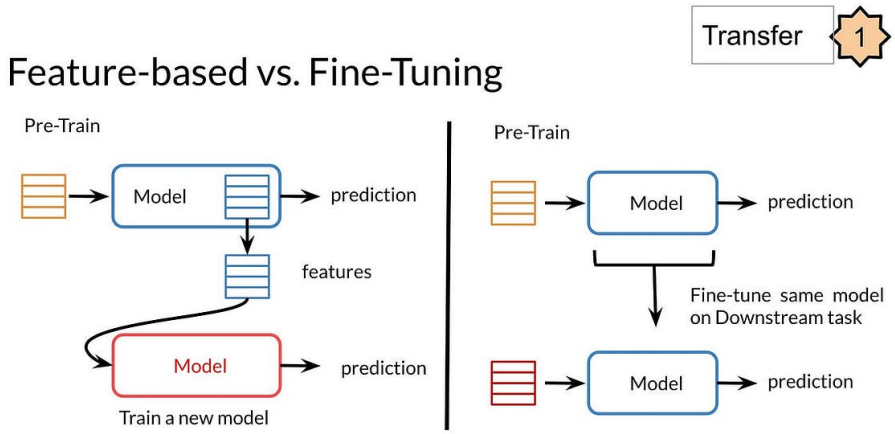


Figura 3.2: Comparativa modelo desde cero vs fine tuned

3.2.2. PyTorch: Una biblioteca para aprendizaje profundo

PyTorch es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca de Torch, utilizado para aplicaciones como visión artificial y procesamiento de lenguajes naturales [38]. Es una biblioteca de código abierto. Esta biblioteca proporciona dos características de alto nivel que la diferencia de otras bibliotecas como TensorFlow:

- Computación de tensores con una aceleración fuerte a través de unidades de procesamientos gráficos (GPU), lo que permite acortar enormemente los tiempos de entrenamiento.
- Redes neuronales profundas construidas en un sistema de diferenciación automática de bases de datos.

A continuación se van a proporcionar las diferentes clases que usaremos de esta biblioteca para construir los diferentes modelos en próximos capítulos.

Dataset [29]

- La clase dataset es una abstracción de tu conjunto de datos, es decir, su función principal es definir cómo debería ser un conjunto de datos. Al ser abstracta tenemos que realizar los siguientes pasos:
- Definir *__init__()*. Este método se encargará de la lectura de datos en csv.
- Reescribir *__len__()*. Este método es el encargado de devolver el tamaño del dataset.
- Reescribir *__get_item__()*. Este método permite la indexación del dataset de manera que se pueda usar dataset[i] para obtener la muestra número i.

A continuación se muestra un ejemplo simple de un dataset con dos columnas.

```
class CustomDataset(Dataset):
    def __init__(self, data):
        self.X = data.loc[:,'X']
        self.y = data.loc[:,'y']

    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]
```

DataLoaders

Los DataLoaders en PyTorch permiten dividir conjuntos de datos en mini lotes automáticamente. En lugar de utilizar todo el conjunto de datos de una sola vez o una sola pieza del conjunto de datos, los DataLoaders utilizan lotes de datos. Esto facilita el entrenamiento de redes neuronales, ya que los modelos entrenan mejor con lotes de datos. La función se compone de tres argumentos

- dataset: Conjunto de datos creado con nuestra clase CustomDataset previamente definida.
- batch_size: tamaño de lote que se desea utilizar.
- shuffle: indica si se desea mezclar los lotes en cada época.

A continuación se muestra un ejemplo de creación de un dataloader con Pytorch:

```
train_loader = torch.utils.data.DataLoader(dataset=dataset, batch_size=batch_size,
    shuffle=False)
```

3.2.3. Fast.ai

Fast.ai es una biblioteca de aprendizaje profundo (deep learning) de código abierto y una plataforma educativa que permite a los desarrolladores y científicos de datos trabajar de manera efectiva con técnicas avanzadas de aprendizaje automático.

Fast.ai está diseñado para hacer que el aprendizaje profundo sea más accesible y fácil de entender para los desarrolladores de todos los niveles de experiencia. Ofrece una API

de alto nivel que simplifica tareas complejas, como el entrenamiento de modelos y la creación de conjuntos de datos. También proporciona una abstracción de nivel superior que permite a los usuarios construir y entrenar modelos de aprendizaje profundo con solo unas pocas líneas de código. Esto facilita la experimentación y la iteración rápida en proyectos de aprendizaje automático.

Esta biblioteca está construida sobre PyTorch, una popular biblioteca de aprendizaje profundo de código abierto. Esto significa que los usuarios pueden aprovechar todas las características y funcionalidades de PyTorch mientras utilizan fast.ai.

Funcionalidades Clave:

- TextBlock: Nos permite la lectura de un conjunto de datos.
- DataBlock: DataBlock es solo un plano sobre cómo ensamblar tus datos. No hace nada hasta que le proporcionas una fuente. Estas fuentes son el TexBlock y las variables dependientes e independientes.
- DataLoaders: Una vez construido el DataBlock se construyen los Dataloaders que va a ser iterados por el bucle de entrenamiento.
- Textclassifierlearner: Permite la creación de un clasificador de texto proporcionando el DataLoader creado y una arquitectura.

A continuación se observa un breve ejemplo de como se crearía una pequeña red dado un conjunto de datos en csv.

```
1
2  dataloaders = DataBlock(TextBlock.from_df('text', is_lm=True),
3  get_x=ColReader('text')).dataloaders(df, bs=128, seq_len=72)
4
5  learn = language_model_learner(
6      dataloaders, AWD_LSTM,
7      opt_func=Adam, loss_func=CrossEntropyLoss,
8      drop_mult=0.3, pretrained=False,
9      metrics=[accuracy])
10 lr = learn.lr_find()
11 learn.fit_one_cycle(10, lr.valley)
```

3.2.4. Hugging Face

Hugging Face 3.3 es una empresa que se ha convertido en un referente en el campo del aprendizaje automático, especialmente en el área de procesamiento del lenguaje natural (NLP, por sus siglas en inglés). Proporcionan una plataforma y una biblioteca de código abierto llamada Transformers que ofrece acceso a una amplia gama de modelos de vanguardia en NLP, así como herramientas y recursos para trabajar con ellos.

La biblioteca de Transformers, la característica principal de Hugging Face, proporciona APIs y herramientas para descargar y entrenar fácilmente modelos preentrenados. Utilizar estos modelos ya entrenados puede ayudar a reducir los costes de computación, ahorro tiempo y ahorro de recursos necesarios para entrenar un modelo desde cero. Estos modelos son compatibles con tareas comunes en diferentes ámbitos, como: procesamiento del Lenguaje Natural, computer vision.

A continuación se puede visualizar la utilización de la biblioteca *Transformers* para cargar un modelo preentrenado de GPT2 [28].

```python
from transformers import GPT2Tokenizer, GPT2LMHeadModel

# Cargar el modelo preentrenado y el tokenizer
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

# Funcion para predecir la siguiente palabra
def predict_next_word(text, max_length=50, num_return_sequences=1):
    inputs = tokenizer(text, return_tensors="pt")
    outputs = model.generate(inputs["input_ids"], max_length=max_length,
        num_return_sequences=num_return_sequences, do_sample=True, top_k=50)
    predictions = [tokenizer.decode(output, skip_special_tokens=True) for output in
        outputs]
    return predictions

# Texto de ejemplo
input_text = "Once upon a time,"
predict_next_word(input_text)
```

Listing 3.1: Ejemplo Transformers Library

A continuación se muestra la entrada y la salida a la función.

Input: Once upon a time
Output: Once upon a time, the war had begun. In the north and southern parts of the continent were several ancient strongholds known to the natives, and by the age of three they had broken loose from the city.
As the battle had ended'

Figura 3.3: Logo Hugging Face

3.2.5. Google Colab

Google Colab, es un entorno de desarrollo integrado (IDE) basado en la nube que permite a los usuarios escribir y ejecutar código Python directamente en el navegador. Google Colab proporciona un entorno de cuadernos Jupyter gratuito y accesible, equipado con capacidades de procesamiento en GPU y TPU, lo que lo convierte en una excelente opción para proyectos que requieren gran capacidad computacional, como para el desarrollo de este proyecto.

Para el desarrollo de este trabajo es la mejor opción ya que nos da acceso a GPUs y TPUs gratuitas. Esto nos permite aprovechar el uso de unidades de procesamiento gráfico (GPU) y unidades de procesamiento tensorial (TPU) para acelerar el entrenamiento de

modelos de aprendizaje profundo.

En la figura se puede ver la GPU que nos proporciona google por defecto de forma gratuita.

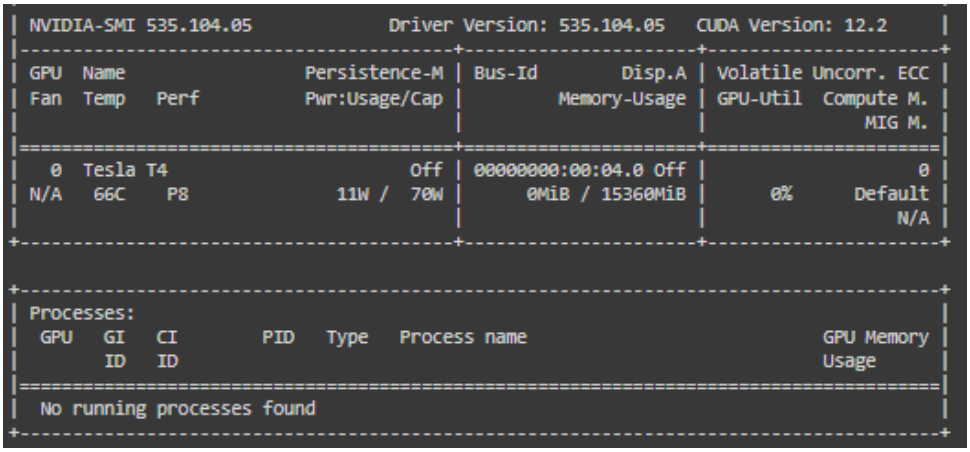


Figura 3.4: Ejemplo GPU Google Colab

Capítulo 4

Planificación del proyecto

Durante este capítulo se explicara la metodología escogida para el desarrollo del proyecto, además de la definición de las diferentes tareas, estimación de esfuerzos, riesgos asociados y estimación de costes.

4.1. Metodología

Tras el análisis de distintos tipos de metodologías, para el desarrollo de este proyecto se ha optado por una metodología iterativa. El ámbito de este proyecto se enfoca en el desarrollo de modelos NLP, por lo que un enfoque iterativo es esencial debido a la naturaleza experimental y progresiva del desarrollo de modelos. A esto hay que sumar la falta de experiencia en el tema a desarrollar, ya que no se conocen lo suficiente las tecnologías como para hacer un diseño a priori. Dado que este trabajo es de aprendizaje e investigación, el modelo de prototipo rápido se ajusta a la perfección, pues nos permite rediseñar continuamente según se vaya consiguiendo un mayor conocimiento de las tecnologías necesarias y según se vaya viendo necesario mejorar en diferentes aspectos. Esta metodología tiene los siguientes aspectos a destacar:

- Adaptabilidad: Permite ajustar y refinar continuamente las estrategias basadas en los resultados obtenidos.
- Incremento de la calidad: Mediante ciclos de prueba y error, se mejora la precisión y robustez de los modelos.
- Flexibilidad: Facilita la incorporación de nuevos datos, herramientas y técnicas a medida que se desarrollan.
- Control de errores: Detecta y corrige errores en etapas tempranas, evitando la propagación de fallos.

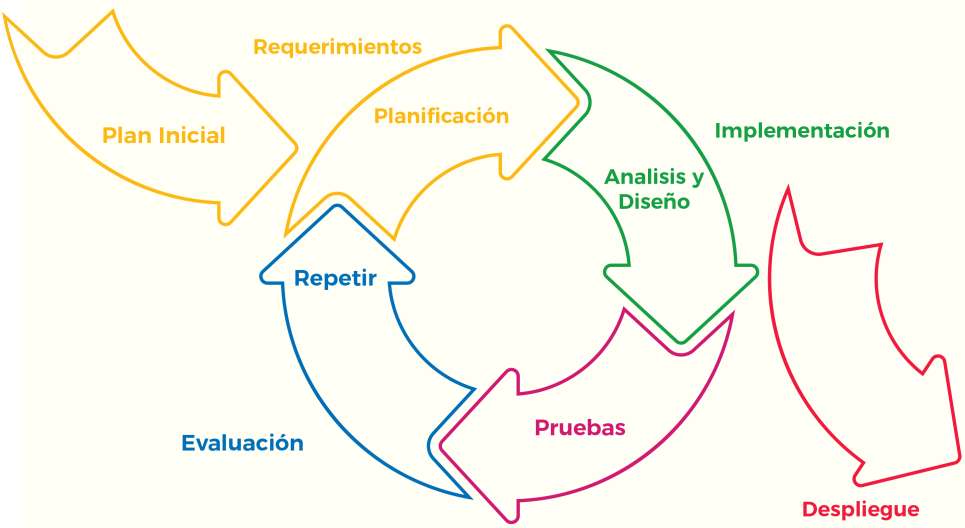


Figura 4.1: Proceso iterativo

4.2. Tareas

Inicialmente el proyecto se divide en cuatro etapas diferenciales, a su vez estas tareas se han dividido en subtareas para un mejor entendimiento del alcance del proyecto.

1. Lectura de papers y documentación. La primera etapa se centra en el estudio del contexto del trabajo y de las herramientas que se van a utilizar para su desarrollo.

   - Lectura de papers proporcionados por el tutor.
   - Entendimiento en profundidad del conjunto de datos.
   - Aprendizaje de las bibliotecas Pytorch y Fast.ai.
   - Desarrollo de la memoria.

2. Utilización del modelo de lenguaje y definición de arquitecturas. Durante esta etapa se utiliza el modelo de lenguaje proporcionado por parte de los tutores del castellano para poder realizar los entrenamientos de las diferentes arquitecturas que utilizaremos.

   - Creación del modelo LSTM para tarea de regresión.
   - Creación del modelo Transformers para la tarea de regresión.
   - Evaluación de los modelos.
   - Desarrollo de la memoria.

3. Creación de las pipelines para la aplicación de técnicas de data augmentation en el conjunto de datos inicial.

   - Creación conjunto de datos mediante label smoothing.
   - Creación conjunto de datos balanceado.
   - Desarrollo de la memoria.

4. Utilización de los nuevos conjuntos de datos creados mediante técnicas de data augmentation para la realización de experimentos sobre los modelos base creados en el punto 2.

    - Creación del modelo LSTM con técnicas de data augmentation para los perfiles y subperfiles.
    - Creación del modelo basado en una arquitectura de transformers con técnicas de data augmentation para los perfiles y subperfiles.
    - Comparativa de resultados de los modelos base y experimentos.
    - Desarrollo de la memoria.

A continuación, en el cuadro 4.1 se presenta los tiempos estimados de cada subactividad.

Tareas	**Horas**
Lectura de documentación	
Lectura de documentación inicial	10
Entendimiento en profundidad del conjunto de datos	20
Aprendizaje de las bibliotecas PyTorch y Fast.ai	30
Desarrollo de la memoria (documentación)	15
Subtotal	**85**
Modelo de Regresión	
Creación del modelo LSTM para tarea de regresión	40
Creación del modelo Transformers para la tarea de regresión	40
Desarrollo de la memoria (documentación)	20
Subtotal	**100**
Ampliación del conjunto de datos data augmentation	
Creación conjunto de datos mediante label smoothing	20
Creación conjunto de datos mediante balanceado	20
Desarrollo de la memoria (documentación)	10
Subtotal	**50**
Experimentación con data augmentation	
Experimento 1 con modelo LSTM	10
Experimento 1 con modelo Transformers	10
Experimento 2 con modelo LSTM	10
Experimento 2 con modelo Transformers	10
Evaluación de los experimentos	15
Revisión y mejora de la documentación de la memoria	10
Subtotal	**65**
TOTAL	**300**

Cuadro 4.1: Estimación de esfuerzo en horas para el proyecto

Tras la definición de la metodología que se va a utilizar para el desarrollo del proyecto, la definición de las diferentes tareas y su estimación de tiempo asociado.Vamos a

definir la implementación de las tareas en el marco de la metodología iterativa mediante la definición de las diferentes iteraciones.

Iteración 1:

- Lectura inicial de documentación.
- Lectura de documentación de las bibliotecas Pytorch y Fast.ai.
- Familiarización con el conjunto de datos a utilizar.
- Estructuración de la memoria y desarrollo de la metodología.

Iteración 2:

- Lectura de documentación asociada a la iteración.
- Inicio de creación modelos base de *LSTM* y *Transformers*.
- Análisis estadístico básico sobre el conjunto de datos.
- Desarrollo de la memoria (Marco teórico).

Iteración 3:

- Lectura de documentación asociada a la iteración.
- Evaluación modelos base de *LSTM* y *Transformers*.
- Desarrollo de la memoria (Conjunto de datos y modelos base).

Iteración 4:

- Creación de los conjuntos de datos (*Label Smoothing*).
- Creación de los conjuntos de datos (Balanceado).
- Escritura de la memoria (Diseño de los experimentos).

Iteración 5:

- Creación de los modelos LSTM y Transfomers para experimento 1.
- Evaluación modelos del experimento 1.
- Escritura de la memoria (Resultados del experimento).

Iteración 6

- Creación de los modelos LSTM y Transfomers para experimento 2.
- Evaluación modelos del experimento 2.
- Escritura de la memoria (Resultados del Experimento).

Iteración	Fecha de Inicio	Fecha de Fin
Iteración 1	2024-03-01	2024-03-16
Iteración 2	2024-03-17	2024-04-01
Iteración 3	2024-04-02	2024-04-20
Iteración 4	2024-04-21	2024-05-01
Iteración 5	2024-05-02	2024-05-20
Iteración 6	2024-05-21	2024-06-05
Iteración 7	2024-06-06	2024-06-16

Cuadro 4.2: Fechas de inicio y fin de cada iteración

Iteración 7

- Revisión y mejora de la memoria.
- Conclusiones y trabajo futuro.

A continuación, se presenta la tabla 4.2 con las fechas asociadas a cada una de las diferentes iteraciones.

Cabe destacar que se produjo un retraso en la Iteración 3, debido a la implementación del modelo base *Transformers*. Esto hizo que el trabajo se produjera un retraso de aproximadamente 12 días. Finalizando la Iteración 8 el 29 de junio.

4.3. Gestión de riesgos

La identificación y prevención de riesgos es algo fundamental que puede afectar al desarrollo del trabajo. Para ello se han identificado riesgos en 5 posibles diferentes categorías de acuerdo a los diferentes tipos de riesgo [5]. Además se han identificado la escala de gravedad de ese riesgo a la vez que la probabilidad de que ocurra. Con esta información se ha detallado con un plan de contingencia acorde al riesgo.

ID	**1a**
Categoría	Riesgos Técnicos
Descripción	La implementación de modelos LSTM y Transformers puede ser más compleja de lo anticipado.
Probabilidad	Media
Impacto	Alto
Plan de contingencia	Buscar ayuda por parte del tutor a la par que realizar una mayor búsqueda en cuanto a la documentación asociada.

Cuadro 4.3: Descripción del riesgo 1a

ID	**1b**
Categoría	Riesgos Técnicos
Descripción	Las técnicas de data augmentation pueden no integrarse bien con los modelos existentes.
Probabilidad	Media
Impacto	Medio
Plan de contingencia	Revisar y ajustar las técnicas de augmentation o considerar alternativas.

Cuadro 4.4: Descripción del riesgo 1b

ID	**2a**
Categoría	Riesgos de Gestión del Proyecto
Descripción	Subestimar el tiempo necesario para completar cada tarea.
Probabilidad	Media
Impacto	Alto
Plan de contingencia	Flexibilizar el cronograma y redistribuir tareas en caso de retrasos.

Cuadro 4.5: Descripción del riesgo 2a

ID	**2b**
Categoría	Riesgos de Gestión del Proyecto
Descripción	Falta de recursos computacionales adecuados para entrenar los modelos.
Probabilidad	Alta
Impacto	Alto
Plan de contingencia	Buscar alternativas de recursos computacionales adicionales o ajustar el alcance del proyecto en función de los recursos disponibles.

Cuadro 4.6: Descripción del riesgo 2b

ID	**3b**
Categoría	Riesgos de Conocimiento y Habilidades
Descripción	Dependencia excesiva del tutor para la comprensión de conceptos clave.
Probabilidad	Media
Impacto	Alto
Plan de contingencia	Búsqueda de los conceptos clave en otras fuentes fiables.

Cuadro 4.7: Descripción del riesgo 3b

4.4. Presupuesto

El presupuesto para el desarrollo de este proyecto se va a dividir en dos categorías: El hardware necesario para llevar a cabo el proyecto y el capital humano.

Hardware Necesario

- Ordenador portátil: Se usará un portátil Asus Zenbook de 2019 para llevar a cabo el desarrollo de modelos, su evaluación y la realización de la memoria. Coste estimado: 1,500 €. Un ordenador de estas características se estima una vida útil entre 4 y 7 años [4].
- GPU (Unidad de Procesamiento Gráfico): necesaria para el entrenamiento de modelos de Machine Learning, especialmente aquellos basados en redes neuronales profundas como LSTM y Transformers. La GPU utilizada es el modelo NVIDIA Testa T4. Tiene un coste estimado de 2,500 € y una vida útil de 5 años.

Capital Humano:

El proyecto requiere de un desarrollador *Machine Learning* para llevar a cabo todas las tareas descritas en el plan del proyecto. El coste anual basándonos en la plataforma Indeed es de 41,463 € [30]. La duración del proyecto de 300 horas teóricas.

A continuación se calculan los costes por hora de cada uno de los elementos del proyecto:

Desglose coste ordenador portátil

Vida útil promedio: (4 años + 7 años) / 2 = 5.5 años
Horas de uso por año (suponiendo 40 horas semanales x 52 semanas): 2,080 horas
Horas de uso totales: 5.5 años x 2,080 horas = 11,440 horas
Costo por hora: 1,500 € / 11,440 horas = 0.13 € por hora

Desglose coste de la GPU:

Vida útil: 5 años
Horas de uso por año: 2,080 horas
Horas de uso totales: 5 años x 2,080 horas = 10,400 horas
Costo por hora: 2,500 € / 10,400 horas = 0.24 € por hora

Desglose coste capital humano

Horas anuales de trabajo: 1,760 (considerando una jornada laboral de 8 horas diarias y 220 días laborables al año).
Coste por hora: 42,604 € / 1,760 horas aproximadamente 23.56 € por hora.
Coste para 300 horas: 23.56 € x 300 horas = 7,068 €.

A continuación, se muestra una tabla con el resumen del presupuesto elaborado, cabe destacar que el presupuesto es aproximado.

Cuadro 4.8: Presupuesto total del proyecto

Elemento	**Costo Total (€)**	**Vida Útil (horas)**	**Costo por Hora (€)**
Ordenador portátil	1,500	11,440	0.13
GPU NVIDIA Tesla T4	2,500	10,400	0.24
Desarrollador ML (300 horas)	7,068	300	23.56
Total General	**11,068**		

Capítulo 5

Modelo del lenguaje

Durante este capítulo se explicará el procedimiento llevado a acabo para obtener el modelo de lenguaje en español y enfocado en el ámbito de las finanzas. Se explicará la metodología utilizada para obtener un modelo de lenguaje en el ámbito financiero apropiado para la tarea de regresión con una arquitectura LSTM. También se explicará el modelo de Transformers *bert-base-multilingual-cased*. Únicamente se aplica *transfer learning* en el modelo LSTM.

Es importante destacar que el modelo de lenguaje no ha sido creada por el autor del TFG, ha sido proporcionada por un colaborador.

Veamos a través de las etapas de la imagen 5.1 los diferentes pasos a seguir para la construcción del modelo de lenguaje. Durante esta sección se explicarán las dos primeras.

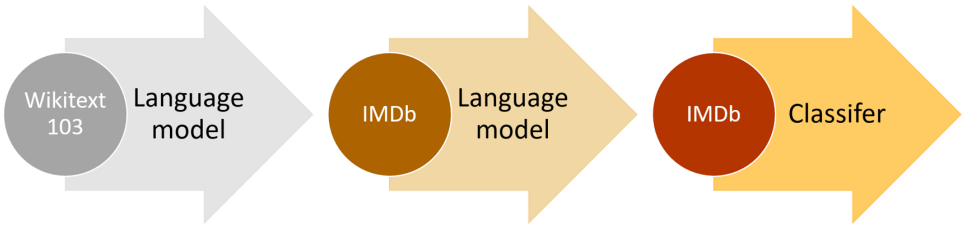


Figura 5.1: Etapas de construcción de la red mediante *transfer learning*

5.1. Modelo LSTM

El modelo LSTMs utilizado para el desarrollo de este trabajo fin de grado es el mismo que el utilizado en el paper [35].

5.1.1. Arquitectura

Para la definición de un modelo de lenguaje primero se debe realizar la tokenización del texto. Para LSTM se utilizan algunos tokens especiales que ayudan controlar aspectos específicos del texto. Estos tokens son fundamentales para garantizar que el modelo procese el texto de manera correcta. A continuación, se describen algunos de los tokens especiales más comunes en *fast.ai*.

- xxbos (*Beginning Of Sentence*): Se coloca al principio de cada nuevo fragmento de texto para indicar el inicio de este.
- xxpad (*Padding*): Se utiliza para rellenar las secuencias de texto hasta una longitud uniforme en un lote (batch).
- xxeos (*End Of Sentence*): Utilizado para marcar el final de una secuencia de texto.
- xxfld (*Field*): Se utiliza para indicar cambios de campo en un conjunto de datos tabular.
- xxunk (*Unknown*): Representa palabras que no están en el vocabulario del modelo.
- xxup (*Uppercase*): Indica que la siguiente palabra está en mayúsculas.
- xxrep (*Repeat*): Indica que la siguiente palabra o carácter se repite varias veces, es decir, si aparece 'xxrep 3 x' indica que la letra "x"se repite 3 veces.
- xxwrep (*"Word Repeat"*): Es similar a *xxrep*, en este caso se hace con palabras completas en lugar de caracteres individuales.

Una vez creado el dataloader con el corpus tokenizado se procede a la creación del *Learner*, esta el la forma en la que la biblioteca *fast.ai* se refiere al modelo. A continuación se va a definir el modelo AWD-LSTM según la biblioteca *fast.ai* [6].

```
AWD_LSTM(vocab_sz, emb_sz, n_hid, n_layers, pad_token=1, hidden_p=0.2, input_p=0.6,
    embed_p=0.1, weight_p=0.5, bidir=False)
```

5.1.2. Entrenamiento

El entrenamiento se divide en dos fases. Una primera fase donde el modelo se entrena en el corpus de la Wikipedia en castellano y una segunda fase donde se realiza *transfer learning* de este modelo sobre diferentes textos y libros financieros.

Primera Fase

Para la primera fase del entrenamiento el modelo de Lenguaje LSTM es fundamental disponer de un corpus extenso y variado que se adapte al problema específico que se desea abordar. Por estos motivos, para este modelo, se ha utilizado una sección aleatoria de la Wikipedia en español [35].

Durante esta primera fase de entrenamiento se entrenó el modelo durante 10 épocas con un *learning rate* de $1 \cdot 10^{-4}$. Usando el optimizador Adam y la función de pérdida de entropía cruzada binaria. A continuación en la figura 5.2 se puede visualizar la función de pérdida para los conjuntos de entrenamiento y validación en esta primera fase de entrenamiento.

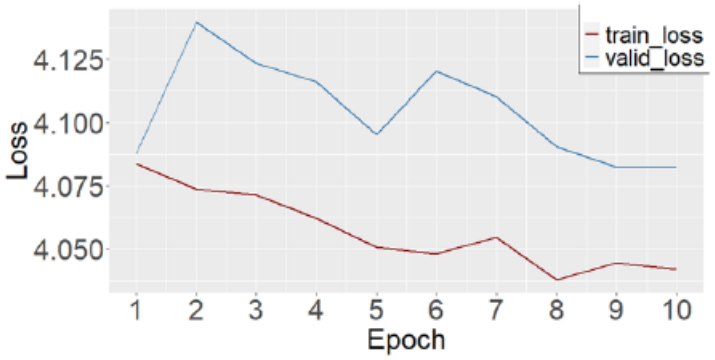


Figura 5.2: Evolución de la función de pérdida para la primera fase entrenamiento (LSTM)

Segunda Fase

En la segunda fase del entrenamiento, se seleccionaron libros académicos de economía y gestión para la realización del *fine tuning* sobre el modelo entrenado en la primera fase. Estos libros no se especifican ya que no se da ningún tipo de información en el paper [35].

El objetivo es que éste capture mejor las particularidades del lenguaje utilizado en estos textos financieros. Al utilizar un corpus con un vocabulario financieramente rico, se busca que el modelo pueda comprender y generar texto que se asemeje al de las biografías de los consejeros, mejorando así su aplicabilidad en contextos relacionados con la economía y la gestión empresarial.

Durante esta segunda fase de entrenamiento se utilizo un *learning rate* $1 \cdot 10^{-3}$ algo mayor que en la primera fase de entrenamiento. En cuanto al resto de hiperparámetros se mantienen igual con respecto a la primera fase del entrenamiento. Se entrena el modelo durante 10 épocas con el optimizador *Adam* y la función de pérdida de entropía cruzada binaria. A continuación en la figura 5.3 se representa la función de pérdida para los conjuntos de entrenamiento y validación en esta segunda fase.

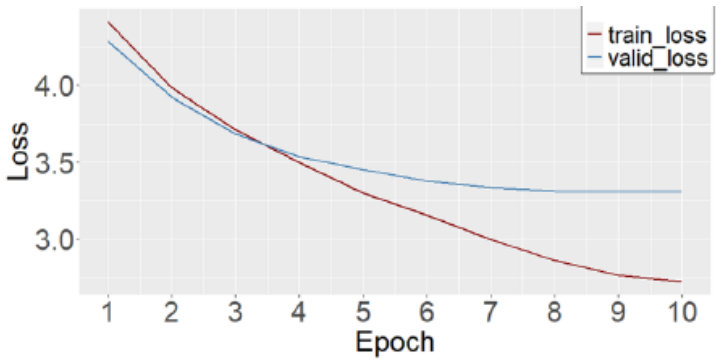


Figura 5.3: Evolución de la función de pérdida para la segunda fase entrenamiento (LSTM)

5.2. Transformers

Para el modelo de lenguaje para el uso de Transformadores se ha optado por la utilización del modelo *bert-base-multilingual-cased* [13]. Este modelo de Bert es una versión de Bert (Bidirectional Encoder Representations from Transformers) entrenada en múltiples idiomas y que distingue entre mayúsculas y minúsculas (cased), se recomienda sobre el uso del modelo *uncased*.

Al ser entrenado sobre el conjunto global de la Wikipedia no se ha visto necesario la realización de un *fine tuning*.

Para la tokenización con el modelo de Bert, los tokens especiales cambian con respecto al modelo *AWD-LSTM*. A continuación, se muestran los diferentes tokens especiales para realizar la tokenización para un modelo Bert.

- **[CLS]** : Este token se añade al inicio de cada secuencia de entrada y su representación final se utiliza a menudo para tareas de clasificación.
- **[SEP]** *Separator*: Se utiliza para separar diferentes secuencias en una entrada.
- **[PAD]** *Padding*: Se utiliza para rellenar las secuencias de texto hasta una longitud uniforme dentro de un lote (batch).
- **[MASK]**: Se utiliza en tareas de modelado de lenguaje enmascarado (MLM), donde se enmascaran algunas palabras en la entrada y el modelo debe predecirlas.
- **[UNK]** *Unknown*: Representa palabras o caracteres desconocidos que no están en el vocabulario del modelo.
- **[BOS]** *Beginning of Sentence*: Es similar a [CLS], pero se usa en algunos modelos para marcar el inicio de una oración.
- **[EOS]** *End of Sentence*: Se encarga de marcar el final de una secuencia de texto. Esto es útil en tareas de generación de texto para indicar cuándo el modelo debe detener la generación.

5.2.1. Arquitectura

Tras la explicación de la tokenización necesaria para un modelo Bert se procede a la explicación de la arquitectura. Bert tiene dos modelos muy similares, pero de diferentes tamaños. Bert Base (12-layer, 768-hidden, 12-heads , 110M parameters) y Bert Large (24-layer, 1024-hidden, 16-heads, 340M parameters) [18]. A continuación se explicará la arquitectura del modelo Base, el utilizado para este proyecto.

A continuación se explican las capas diferentes con respecto a otras arquitecturas:

- *Token Embeddings*: Cada token en la secuencia de entrada se convierte en un vector de dimensión fija. Estos vectores se obtienen a partir de una tabla de embeddings que se entrena junto con el modelo BERT.

- *Positional Embeddings*: Se utiliza una codificación posicional, similar a la utilizada en la red transformer original. La codificación posicional es necesaria para indicarle al bloque de codificación la posición relativa de cada palabra dentro de la frase, ya que todas las palabras son procesadas de manera simultánea por la red.
- *Segment Embeddings*: Se añade un embedding que indica a qué segmento pertenece la frase: la primera frase (antes del separador) se codifica con un embedding diferente al de la segunda frase.
- Capas que representan las consultas y las parejas clave valor.

A continuación en la figura 5.4 se muestra un ejemplo del proceso de la formación de los *embeddings* con la frase "Mi gato juega y luego duerme".

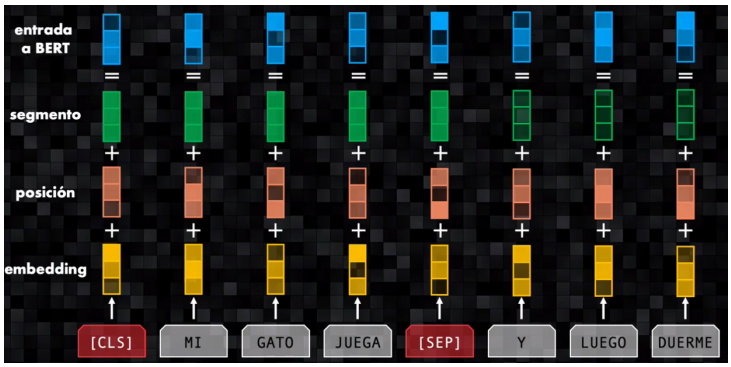


Figura 5.4: Bert Embeddings

Capítulo 6

Regresión

Durante este capítulo se detalla el proceso de construcción del modelo de regresión para los CV de los consejeros independientes.

6.1. Conjunto de datos

El conjunto de datos consta de 137 empresas que cotizan en el Mercado Continuo de la Bolsa de Madrid. Se han excluido aquellas pertenecientes al sector financiero. El estudio abarca el período 2003-2020. Se analizaron un total de 6561 perfiles de directores. La mayoría de las empresas no permanecieron durante todo el período, sino que fueron excluidas en un momento dado o se incorporaron sucesivamente.

La descripción de los perfiles de los directores (CVs) se tomó de los Informes Anuales de Gobierno Corporativo de las empresas cotizadas, los cuales se pueden consultar en formato pdf en la página web de la Comisión Nacional del Mercado de Valores (CNMV). Con el paso de los años, se puede observar que la descripción de los perfiles se ha ido ampliando. En los primeros años, las descripciones eran muy breves o simplemente inexistentes. Asimismo, existen ciertas diferencias entre las empresas en el grado de detalle de dichas descripciones.

De los 6561 perfiles de directores, se han seleccionado 1023 para obtener un conjunto de entrenamiento sobre el cual se realiza un etiquetado manual, necesario para la etapa de entrenamiento supervisado del modelo de regresión. El criterio para seleccionar el conjunto de entrenamiento es una distribución uniforme entre los diferentes tipos de perfiles. De estos 1023 perfiles, se han separado alrededor de 100 biografías, elegidas aleatoriamente, para obtener un conjunto de test que no participa en el entrenamiento y que permite realizar una validación de los resultados. Además, el conjunto de entrenamiento se divide internamente en dos grupos para realizar un entrenamiento estándar con una fase de validación al final de cada época.

El experto humano ha etiquetado cada perfil de director del conjunto de datos de entrenamiento utilizando la información de su biografía. Para cada uno de los seis perfiles y ocho subperfiles, se asigna un número entre 0 y 1 usando un solo dígito decimal, es decir, se realiza una evaluación estándar entre 0 y 10 que estima el peso del perfil correspondiente dentro del CV. Estos perfiles no se han considerado mutuamente excluyentes, por lo que se pueden obtener valores máximos (1.0) en múltiples e incluso en todos los

perfiles. Por esta razón, el criterio utilizado en el etiquetado consiste en evaluar los datos del CV de manera equilibrada entre los diferentes directores y no realizar una asignación porcentual. Esto permite una evaluación más objetiva y comparable entre los diferentes directores. Dado que la fuente de los datos es un CV en forma de texto libre no estructurado y en el que la expresión de méritos similares puede provenir de expresiones escritas con estructuras y vocabulario muy diferentes, el etiquetado debe interpretarse como un hecho indicativo del perfil profesional y no como un valor numérico exacto.

6.1.1. Análisis del conjunto de datos

El conjunto de datos, como se ha mencionado anteriormente, no lo he obtenido personalmente, la extracción se ha realizado mediante un proceso ETL y etiquetado manualmente por un experto, tal y como se ha especificado anteriormente. Tras la realización de estas labores se ha obtenido el conjunto de datos con las siguiente variables:

- Variables Dependientes(Categorías)
  - Categorías Principales
    - F: Financiero
    - E/C: Directivo/Consultor
    - A/T/A: Auditor/Contable/Fiscal
    - L: Legal
    - P: Político (Incluye puestos en instituciones)
    - AC: Académico
  - Categorías Ampliadas
    - FB Financiero bancario
    - FnB Financiero no bancario
    - CEO Consejero Delegado y similar
    - E/C_E Directivo Estrategia
    - E/C_M Directivo marketing
    - E/C_RH Directivo RRHH
    - E/C_I Directivo Internacional
    - E/C_F Directivo Finanzas (Dtor. Financiero...)
- Variables Independientes
  - bio: Texto con la biografía del individuo
  - empresa: Empresa al que pertenece el consejero
  - nif: Código de identificación tributaria.
  - fecha

Es importante destacar que las 8 categorías ampliadas son categorías de dos de las variables principales (F y E/C). Los perfiles financieros se pueden dividir a su vez en FB (Financiero Bancario) y FnB (Financiero no bancario). Mientras que el perfil principal

E/C se puede subdividir en el resto de categorías ampliadas.

Una vez dada esta aclaración, se van a crear dos modelos. El primero será para la las categorías principales y el otro para las subcategorías. El modelo va a predecir las variables independientes que han sido etiquetadas por el profesor doctor en economía financiera D. Fernando Tejerina. El etiquetado se realiza sobre un subconjunto de las biografías y se utilizará para entrenar la red neuronal con el objetivo de predecir el resto de valores si los resultados sobre el subconjunto son buenos. Son puntuaciones que varían en el rango [0, 1] dependiendo del nivel de experiencia del consejero en cada campo, donde un nivel más cercano a 1 se considera un nivel mayor.

6.1.2. Exploración del conjunto de datos

Vamos a realizar una exploración exhaustiva de los datos con el objetivo de obtener una comprensión detallada y clara de su estructura y características. Este análisis descriptivo previo es crucial porque nos permite obtener más información sobre los diferentes perfiles y subperfiles que podrían afectar el rendimiento y la precisión de los modelos.

En primer lugar se ha realizado un análisis de frecuencia de palabras en las biografías. Contamos con un conjunto de 6561 biografías en el conjunto de datos. Las biografías cuentan con 105221 palabras en total, de las cuales 10105 son diferentes. En la figura 6.1 se ha realizado un histograma con las palabras más frecuentes de las biografías de los consejeros independientes. Se han eliminado para esta representación palabras conectoras, preposiciones, etc. Es decir, palabras con carencia de significado en el contexto.

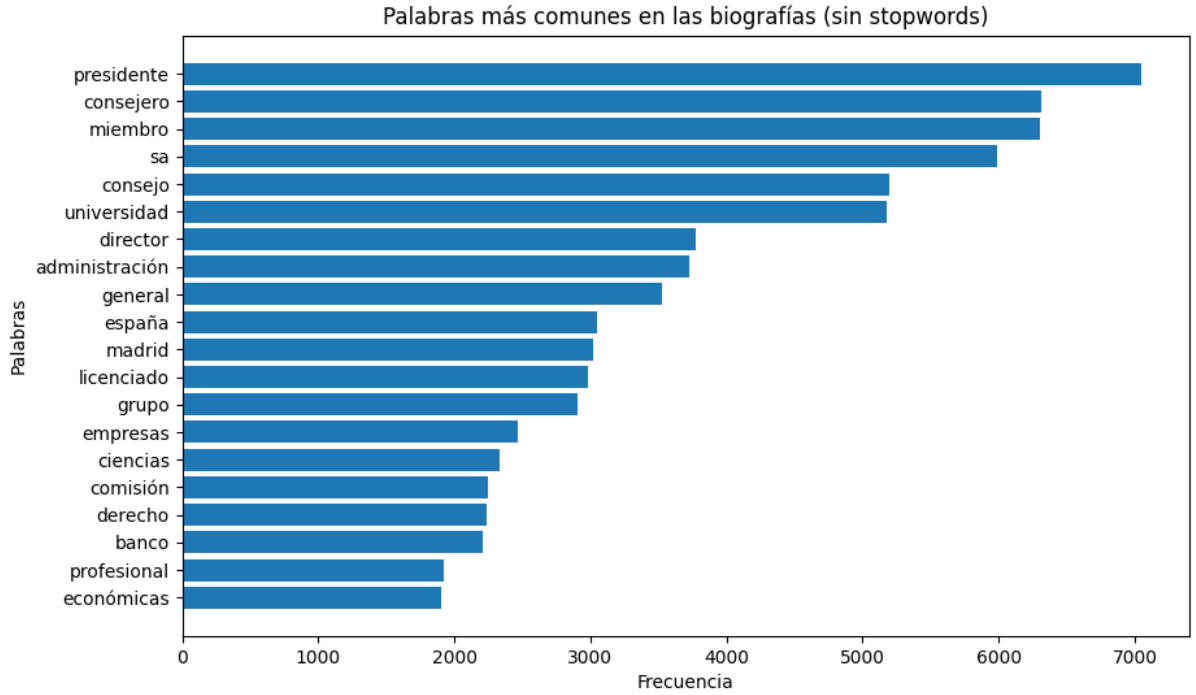


Figura 6.1: Histograma palabras más comunes en el conjunto de datos

Observamos en la figura 6.1 que las palabras más comunes se relacionan con términos de juntas directivas, estudios y cargos importantes en una compañía, es decir, términos

financieros.

En segundo lugar, se ha realizado un estudio sobre las categorías principales y subcategorías del conjunto de datos. En la figura 6.2 podemos ver la frecuencia de aparición de las categorías principales y subcategorías, es decir, las veces que un perfil/subperfil ha sido etiquetado con un valor mayor que 0.

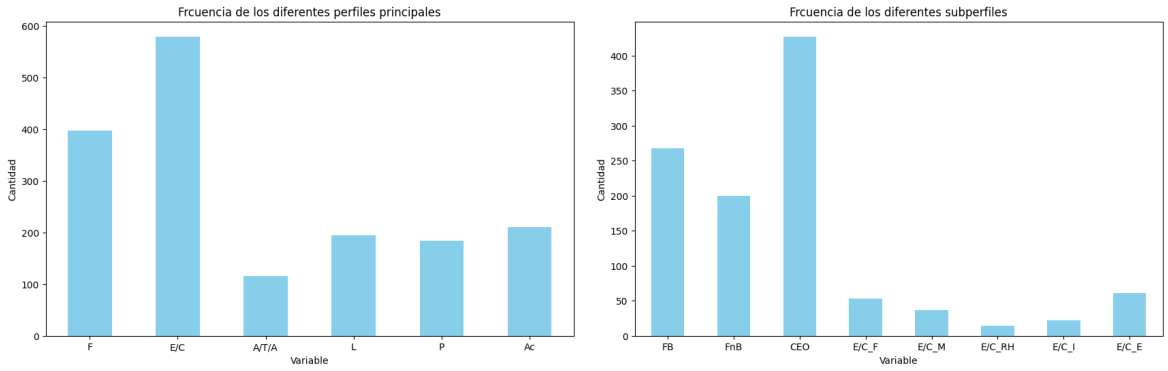


Figura 6.2: Histograma frecuencia (perfiles y subperfiles)

Observamos que los 6 perfiles y 8 subperfiles no están distribuidos de forma homogénea, es decir, el conjunto de datos no está balanceado. Esto afectará a los modelos en su proceso de generalización de la información y perjudicando a las clases menos representadas.

En la tabla 6.1 se presenta un resumen estadístico descriptivo de los seis perfiles principales (F, E/C, A/T/A, L, P y Ac).

	count	mean	std	min	25 %	50 %	75 %	max
F	1023	0.281	0.400	0.000	0.000	0.000	0.600	1.000
E/C	1023	0.525	0.475	0.000	0.000	0.700	1.000	1.000
A/T/A	1023	0.096	0.282	0.000	0.000	0.000	0.000	1.000
L	1023	0.166	0.358	0.000	0.000	0.000	0.000	1.000
P	1023	0.109	0.272	0.000	0.000	0.000	0.000	1.000
Ac	1023	0.161	0.340	0.000	0.000	0.000	0.000	1.000

Cuadro 6.1: Estadísticas descriptivas de las categorías principales

En la tabla 6.1 observamos que las categorías muestran medianas cercanas al 0. Esto nos indican una distribución sesgada hacia valores bajos con algunos valores extremos, es decir, las etiquetas no suelen representar valores intermedios, es más una clasificación de pertenencia al perfil. Lo mismo ocurre en las subacategrías, se puede observar en la tabla 6.2.

	count	mean	std	min	25 %	50 %	75 %	max
FB	1023	0.169	0.325	0.000	0.000	0.000	0.200	1.000
FnB	1023	0.108	0.251	0.000	0.000	0.000	0.000	1.000
CEO	1023	0.365	0.454	0.000	0.000	0.000	1.000	1.000
E/C_F	1023	0.032	0.152	0.000	0.000	0.000	0.000	1.000
E/C_M	1023	0.016	0.092	0.000	0.000	0.000	0.000	1.000
E/C_RH	1023	0.007	0.070	0.000	0.000	0.000	0.000	1.000
E/C_I	1023	0.011	0.084	0.000	0.000	0.000	0.000	1.000
E/C_E	1023	0.035	0.156	0.000	0.000	0.000	0.000	1.000

Cuadro 6.2: Estadísticas descriptivas de las subcategorías

6.2. Modelo LSTM

Durante esta sección se va explicar los modelos creados con una Arquitectura LSTM a partir del modelo de lenguaje financiero explicado en la sección anterior tanto para las categorías principales como para las subcategorías.

Vamos a utilizar *transfer learning* con *fast.ai* para el modelo de lenguaje financiero proporcionado, el cual hemos explicado en el capítulo anterior, para realizar la regresión.

El modelo de lenguaje se basa en predecir las siguientes palabras de una cadena. Mientras que nuestro objetivo es proporcionar una predicción para cada una de las diferentes categorías. Para ello tendremos que realizar modificaciones y añadir unas capas.

En cuanto a su arquitectura tenemos que seguir la misma el cual el modelo ha sido entrenado para los textos financieros. Usaremos una arquitectura AWD-LSTM por defecto que tiene el modelo de la biblioteca *fast.ai*. Está tiene las siguientes capas:

- Normalización
- Dropout
- Linear
- ReLu
- Normalización
- Dropout
- Capa Lineal
- Función Sigmoidea de activación

En la figura 6.3 se puede ver una descripción de la red para las categorías principales y subcategorías.

Cabe destacar, que la arquitectura del modelo es común tanto para la regresión de las categorías principales como para la regresión de las subcategorías.

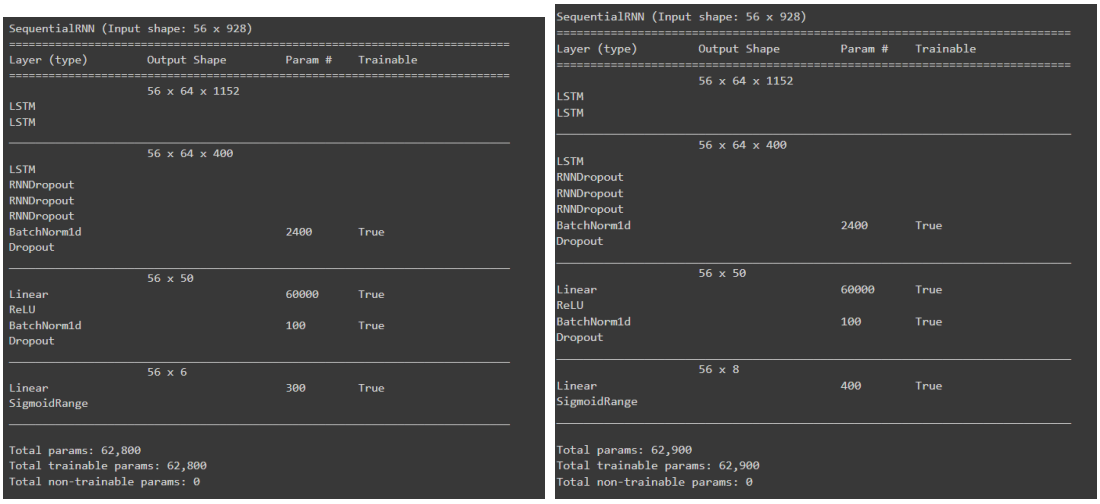


Figura 6.3: Resumen arquitectura redes AWD_LSTM para categorías principales y subcategorías

6.2.1. Conjuntos de datos para perfiles

Debido a la poca cantidad de datos etiquetados, apenas unos mil, 1023 biografías etiquetadas exactamente. Es crucial mantener la mayor cantidad de datos para el entrenamiento. Por eso en vez de optar por la clásica división 70 % y 30 %, se ha optado por una división que favorezca al conjunto de entrenamiento 90 %, 10 %. Además se toman un 10 % del conjunto train para el conjunto de validación. Para la división del conjunto de datos en las categorías principales entre train y test se ha utilizado la función *train_test_split* de *sklearn.model_selection*.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
    random_state=42)
df_train = df.iloc[X_train.index]
df_test = df.iloc[X_test.index]
```

A continuación, se crean los dataloaders mediante la utilización de la biblioteca *fast.ai*. Aquí se deja implícita la división del conjunto de entrenamiento y el de validación.

```
def get_y(r): return (tensor([r[col] for col in columnas_y]))

# Define a DataBlock for text data
text_datablock = DataBlock(
    blocks= (TextBlock.from_df('bio',vocab=vocab), RegressionBlock(n_out=6)),
    get_y = get_y,
    get_x=ColReader('text'), # Extracts the text data for model input
    splitter=RandomSplitter(valid_pct = 0.1, seed = 81) # Split the data into training
        and validation sets (10% validation)
)
text_datablock = text_datablock.dataloaders(df_train, bs=56, seq_len=80)
```

6.2.2. Conjuntos de datos y dataloader para subperfiles

La metodología empleada para la división del conjunto de datos se ha mantenido. Se ha aplicado una proporción del 90 % para entrenamiento y 10 % para prueba. La misma proporción se ha utilizado para dividir el conjunto de designado para el entrenamiento

en dos (entrenamiento y validación).

Para la creación del conjunto de prueba, se desarrolló una función propia con el objetivo de asegurar una representación equilibrada de todas las categorías presentes en el conjunto de datos.

```

def create_balanced_test_set(df):
    # Anadir una columna que identifique las categorias
    def categorize(row):
        if row.sum() == 0:
            return 'all_zero'
        else:
            return row.idxmax()

    df['category'] = df.apply(categorize, axis=1)

    # Crear un DataFrame para almacenar el conjunto de test
    test_set = pd.DataFrame(columns=df.columns)

    # Seleccionar 10 observaciones por categoria
    categories = df['category'].unique()
    for category in categories:
        category_df = df[df['category'] == category]

        if len(category_df) >= 10:
            test_samples = category_df.sample(n=10, random_state=1)
        else:
            test_samples = category_df.sample(n=len(category_df), random_state=1)

        test_set = pd.concat([test_set, test_samples])

    # Eliminar la columna de categoria antes de devolver el conjunto de test
    test_set = test_set.drop(columns=['category'])

    return test_set

# Ejemplo de uso:
# df es tu DataFrame original con las 8 categorias y valores entre 0 y 1
test_set = create_balanced_test_set(df_train.iloc[:,1:])

df_test = df_train.iloc[test_set.index,:]
positions_to_exclude = [x - 1 for x in test_set.index]
df_train = df_train.drop(positions_to_exclude)
```

Listing 6.1: Example Transformers Library

El dataloader se ha implementado utilizando el mismo código que para la regresión de las categorías principales. La única modificación realizada ha sido la adaptación en el número de etiquetas en la salida de regresión para las 8 subcategorías.

Entrenamiento

El modelo se entrena durante 250 épocas utilizando un optimizador Adam y una función de pérdida MSE (Error Cuadrático Medio), ya que estamos abordando un problema de regresión y no de categorización. Se establece un learning rate base de $2e-3$, determinado como óptimo tras la aplicación de la función *learnlr_find()*.

Para el entrenamiento, se realiza un congelamiento y descongelamiento automático de capas mediante la función *learnfine_tune()* de *fast.ai*. A continuación se muestran la figura 6.4 que representan la evolución de la función de pérdida para el conjunto de entrenamiento y validación, tanto para categorías principales como para las subcategorías.

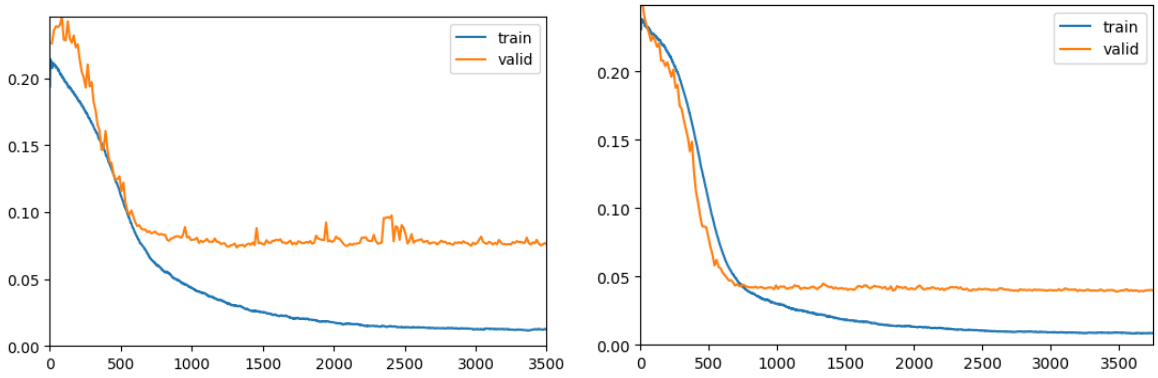


Figura 6.4: Evolución de la función de pérdida para el entrenamiento (AWD-LSTM)

Métricas de evaluación

En las siguientes subsecciones se utilizan tres métricas para evaluar los resultados sobre el conjunto de test:

- MSE (Error Cuadrático Medio) [39]: Es un estimador que mide el promedio de los errores al cuadrado, es decir la diferencia entre la estimación y el valor real. Si $\hat{Y}$ es un vector de n predicciones y Y es el vector de los verdaderos valores, entonces se define el MSE como
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

- MAE (Error Medio Absoluto): Mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección. Es la diferencia absoluta promedio entre los valores predichos y los valores reales. A diferencia del MSE, no eleva al cuadrado los errores, lo que significa que no penaliza los errores más grandes tan severamente.

- R2: Se trata de la proporción de variabilidad explicada de la variable dependiente por la variable independiente, en nuestro caso, las variables independientes son las biografías y las variables dependientes son los perfiles y subperfiles. Toma valores entre 0 y 1. Tomara un valor de 0 cuando proporcionada una salida constante. Por otro lado, tomara un valor de 1 si las predicciones coinciden con los valores reales. La métrica R2 se expresa de la siguiente forma:
$$1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

6.2.3. Validación perfiles

Tras el entrenamiento del modelo para el conjunto de entrenamiento recordemos que se reservo un 10 % de observaciones para el conjunto test. Más concretamente en este conjunto test encontramos 103 biografías. En la tabla 6.3 tenemos los resultados usando las métricas de evaluación mencionadas en la sección anterior.

- MSE: Encontramos un valor medio de 0.1042. Es de destacar los valores de los perfiles A/T/A, L y P que han obtenido valores bajos respecto al resto. Este resultado indica que para estos tres perfiles el modelo tuvo mayor precisión.

Métricas	F	E/C	A/T/A	L	P	Ac	Media
MSE	0.1186	0.1913	0.0645	0.0548	0.0471	0.1491	0.1042
MAE	0.2019	0.2844	0.0926	0.0943	0.1036	0.2015	0.1630
R2	0.3365	0.1555	0.3785	0.4983	0.4163	0.4897	0.3791

Cuadro 6.3: Métricas Evaluación (MSE, MAE, R2) AWD-LSTM perfiles (Base)

- MAE: Tenemos un MAE medio de 0.1630. Esto nos indica que la predicción total del modelo se desvía de la realidad un 16.3 %.
- R2: Este modelo de categorías principales explica aproximadamente un 38 % de la variabilidad total de los datos. Este valor es relativamente bajo. La media se ve afectada por la categoría de consultor. En los próximos capítulos se aplicaran técnicas de *data augmentation* para ver si obtenemos una mejora de la variabilidad explicada por el modelo.

Para una validación más exhaustiva se propone una validación en dos partes. En la primera, se lleva a cabo una modelización numérica del modelo utilizando datos etiquetados que la red no ha utilizado para el entrenamiento, es decir, el conjunto test. De esta manera, es posible medir cómo este modelo imita el comportamiento del experto humano. En la segunda etapa, se diseña un escenario de cuantificación para asignar cada uno de los seis perfiles a una categoría y medir el grado de precisión con el que nuestro modelo de DL identifica el perfil de un director.

La primera parte de la validación se puede observar en la tabla 6.4. los perfiles L y P muestran los mejores indicadores de desempeño en términos de baja variabilidad y alta precisión. El perfil Ac, por otro lado, muestra la mayor variabilidad y menor precisión, lo que sugiere que el modelo tiene más dificultades para predecir con precisión este tipo de perfil.

Perfil	μ (error)	σ (error)	Corr. Coef. (R)	Std. Error of μ
F	-0.051958	0.340459	0.625201	0.033546
E/C	0.036557	0.435852	0.519401	0.042946
A/T/A	-0.056624	0.247550	0.643599	0.024392
L	-0.044722	0.229728	0.719449	0.022636
P	-0.036198	0.214091	0.658230	0.021095
Ac	0.083404	0.376973	0.371024	0.037144

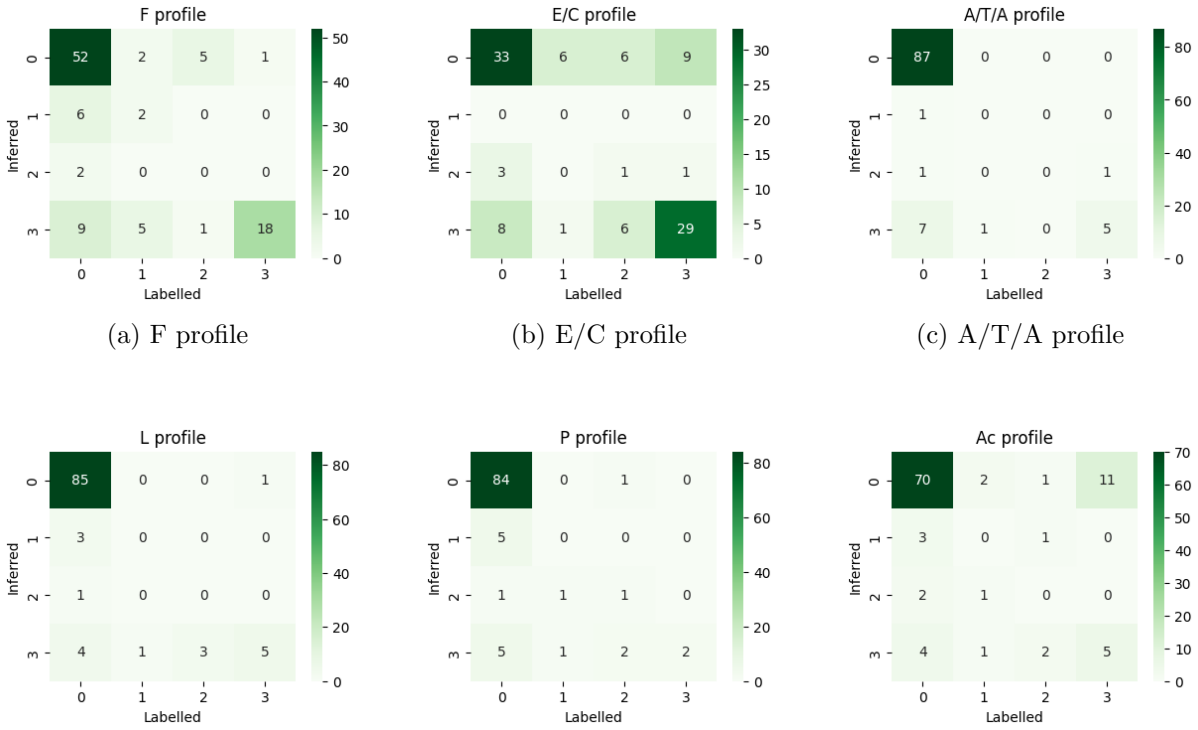
Cuadro 6.4: Validación modelo AWD-LSTM perfiles (Base)

Vistos los resultados de la tabla 6.4 observamos la naturaleza imprecisa de los datos utilizados para entrenar la red, sumándole que el etiquetado realizado por el experto humano es, por la misma razón, aproximado, obtener datos exactos de nuestro modelo no puede considerarse como el objetivo principal del mismo. Sin embargo, es posible asignar una categoría a cada uno de los seis perfiles para aproximar su evaluación utilizando un número menor de intervalos. Se han realizado 4 categorías atendiendo a los siguientes intervalos. [0,0.3), [0.3, 0.5), [0.5, 0.7) y [0.7,1.0] que corresponden con las categorías 0,1,2,3 respectivamente. En la tabla 6.5 se muestra la tasa de aciertos de la red cuando se produce esta discretización de los valores con una media de acierto del 77.5 %.

Perfil	F	E/C	A/T/A	L	P	Ac
Test	69.903	61.165	89.32	87.3786	84.466	72.815

Cuadro 6.5: Precisión modelo AWD-LSTM perfil (Base)

En la figura 6.5 observamos la matriz de confusión para los 6 diferentes perfiles para el conjunto de test. Podemos observar que las categorías centrales (1,2) están mucho menos representadas en comparación con las categorías (0,3). Como ya se mencionó en el análisis descriptivo de los datos, los perfiles suelen ser o todo o nada, mientras que los valores intermedios suelen ser poco frecuentes.



(a) F profile (b) E/C profile (c) A/T/A profile

(d) L profile (e) P profile (f) Ac profile

Figura 6.5: Matriz confusión perfiles modelo AWD-LSTM

En los anexos se pueden encontrar las predicciones realizadas por los diferentes modelos, tanto para perfiles como subperfiles, para diferentes biografías del conjunto test.

6.2.4. Validación subperfiles

Tras el entrenamiento del modelo AWD-LSTM con la biblioteca *fast.ai* para las 8 subcategorías, se ha realizado una validación con un conjunto de test con un total 87 biografías. La obtención de este conjunto de test se ha explicado en secciones anteriores. La evaluación y validación de este modelo ha seguido la misma estructura que la sección anterior.

En la tabla 6.6 podemos encontrar los resultados obtenidos para los 8 subcategorías con las métricas de evaluación definidas.

Métrica	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**	**Media**
MSE	0.0106	0.0105	0.0798	0.0267	0.0543	0.0489	0.0267	0.0661	0.0405
MAE	0.0546	0.0488	0.1290	0.0482	0.0815	0.0641	0.0587	0.1118	0.0746
R2	0.8898	0.8206	0.0610	0.6912	0.0246	0.0846	0.5193	0.2246	0.4144

Cuadro 6.6: Métricas Evaluación (MSE, MAE, R2) AWD-LSTM subperfiles (Base)

- MSE: El MSE varía significativamente entre las subcategorías, con valores que oscilan entre 0.0105 y 0.0798. La media del MSE es 0.0405, lo cual es un valor moderadamente bajo, sugiriendo que, en promedio, las predicciones del modelo no se desvían considerablemente de los valores reales.
- MAE: El MAE comprende valores de desde 0.0482 hasta 0.1290. La media del MAE es 0.0746, lo que indica en general que se prodcue una desviación de 7.46 % de los valores etiquetados.
- R2: El R2 presenta una media de 0.4144, es un valor aceptable aunque bajo. Observando las categorías vemos que la mayor variabilidad la explican los perfiles financieros tanto bancarios como no bancarios. Obteniendo un R2 muy bajo en categorías como CEO, E/C_M. Categorías que se tendrán en cuenta en capítulos posteriores al hacer un aumento de datos.

Como se ha desarrollado en la sección anterior, en primer lugar vamos a realizar una medición donde se imita el comportamiento humano. La validación del modelo que imita el comportamiento de etiquetado del humano se puede ver en la tabla 6.7.

El análisis de la tabla que representa lo bien que nuestro modelo imita la clasificación por categorías de un experto humano revela varias características importantes. Primero, los perfiles financieros muestran los mejores resultados con errores medios (μ) muy cercanos a cero y coeficientes de correlación (R) superiores al 90 %, indicando una fuerte concordancia con las clasificaciones del experto humano. Además, estos perfiles tienen desviaciones estándar (σ) bajas, esto sugiere una baja variabilidad en los errores y una alta consistencia en las predicciones del modelo para estas categorías.

Por otro lado, los perfiles E/C_M y E/C_RH presentan los errores medios más altos, con coeficientes de correlación (R) bajos. Esto sugiere que, aunque el modelo LSTM es generalmente eficaz, existen diferentes perfiles donde el modelo no es capaz de captar los

Subperfil	μ **(error)**	σ **(error)**	**Corr. Coef. (R)**	**Std. Error of** μ
FB	-0.010002	0.102614	0.943856	0.011001
FnB	0.005949	0.102272	0.909978	0.010965
CEO	0.057455	0.276570	0.629256	0.029651
E/C_F	-0.026007	0.161402	0.836390	0.017304
E/C_M	-0.066648	0.223231	0.323091	0.023933
E/C_RH	-0.059954	0.212916	0.120274	0.022827
E/C_I	-0.008632	0.163151	0.727647	0.017492
E/C_E	-0.078730	0.244767	0.555721	0.026242

Cuadro 6.7: Validación modelo AWD-LSTM subperfiles (Base)

patrones necesarios para un buen ajuste en esos perfiles.

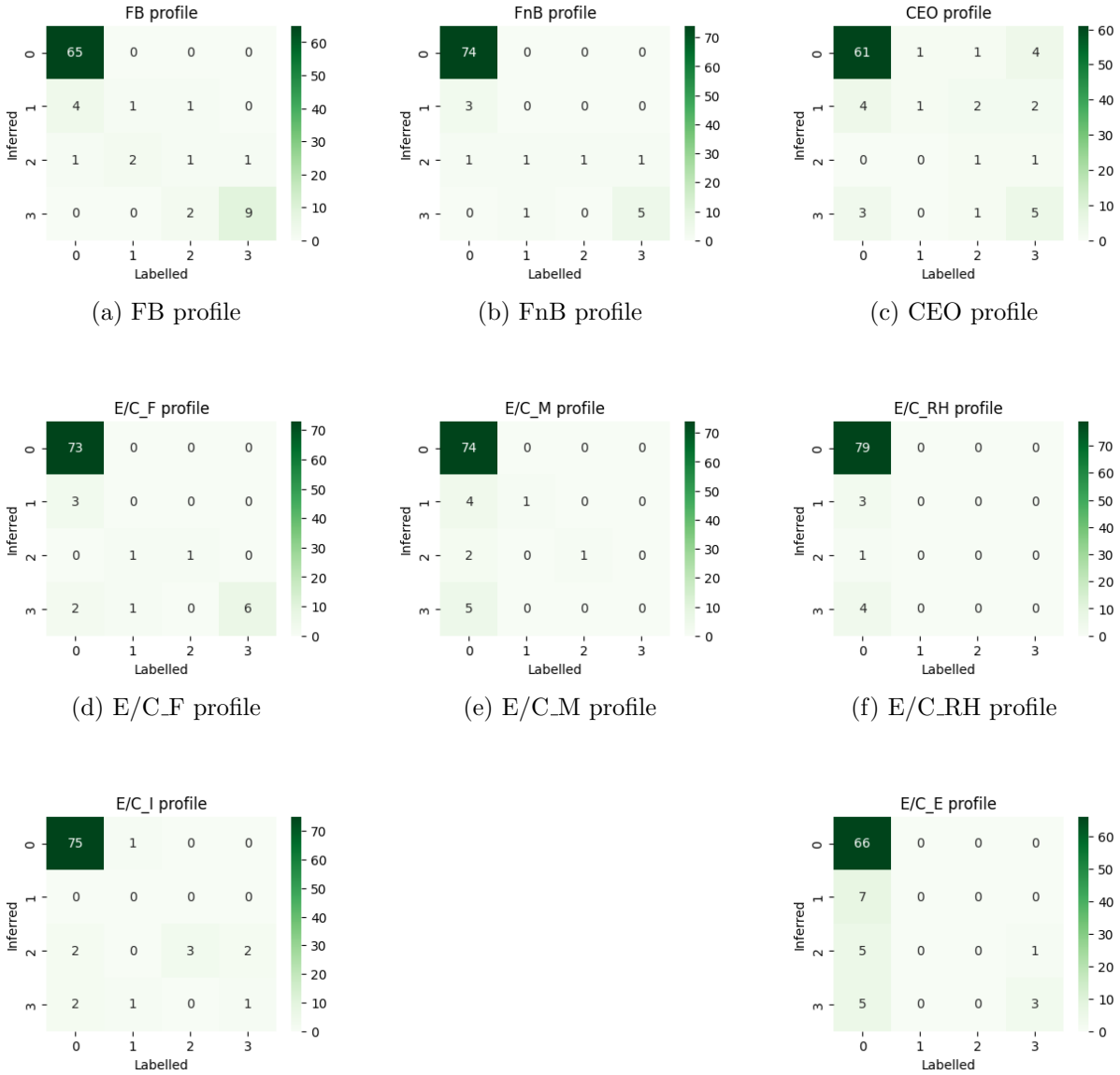
Procedemos con la segunda parte de la validación. Como ya se mencionó en la sección anterior vamos a realizar una discretización de los valores en 4 categorías en los siguientes intervalos [0, 0.3), [0.3, 0.5), [0.5, 0.7) y [0.7,1.0] que corresponden con las categorías 0,1,2,3 respectivamente. De esta forma podemos medir el grado de precisión en el que nuestro modelo identifica las diferentes subcategorías.

El calculo de la precisión para las 8 subcategorías discretizadas se encuentra en la tabla 6.8. Tenemos una precisión media del 87.21 %.

Subperfil	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**
Test	87.356	91.954	78.161	91.95	87.356	90.804	90.805	79.31

Cuadro 6.8: Precisión modelo AWD-LSTM subperfil (Base)

A continuación, en la figura 6.6 las matrices de confusión de los diferentes perfiles con sus valores discretizados. Observamos como ocurre lo mismo que el los perfiles principales. Los valores tienden a concentrarse en los extremos.



(a) FB profile (b) FnB profile (c) CEO profile

(d) E/C_F profile (e) E/C_M profile (f) E/C_RH profile

(g) E/C_I profile (h) E/C_E profile

Figura 6.6: Matriz de confusión modelo AWD-LSTM subperfiles

6.3. Transformers

Este modelo se ha realizado con la librería *Pytorch* en vez de la librería de *fast.ai*. Para no caer en la repetición de los mismos conceptos durante el desarrollo de esta sección se explicaran únicamente las diferencias en la implementación con los modelos con arquitecturas LSTM.

Al igual que la sección anterior es necesario modificar las últimas capas del modelo Bert para nuestra tarea de regresión.

6.3.1. Conjunto de datos y dataloaders

Para la creación del dataloaders se han utilizado tamaños de lote diferente. Se ha utilizado un *batch_size* de 64 para el conjunto de entrenamiento, mientras que para el conjunto de validación y test se ha usado un *batch_size* de 256. Este cambio en el tamaño de lote afectará al número de épocas para los que entrenaremos el modelo.

6.3.2. Arquitectura

Se ha añadido la misma estructura de capas que en el modelo AWD-LSTM. Con el objetivo de que ambos modelos sean idénticos para favorecer una comparativa más objetiva. En la tabla 6.9 se puede ver un resumen del modelo de Transformers para los 6 perfiles principales.

Capa	**Dimensión**	**Repeticiones**	**Núm. Parámetros**
Embedding de palabras	119547 x 768	1	91876320
Embedding de posiciones	512 x 768	1	393216
Embedding de segmentos	2 x 768	1	1536
Normalización	768	1	1536
Dropout	-	1	0
Consulta	768 x 768	12	589824 x 12
Clave	768 x 768	12	589824 x 12
Valor	768 x 768	12	589824 x 12
Dropout	-	12	0
Atención (lineal)	768 x 768	12	589824 x 12
Normalización	768	12	1536 x 12
Intermedia (lineal)	768 x 3072	12	2359296 x 12
Normalización	768	12	1536 x 12
Dropout	-	12	0
Salida (lineal)	3072 x 768	12	2359296 x 12
Normalización	768	12	1536 x 12
Dropout	-	12	0
Pooler (lineal)	768 x 768	1	589824
Activación (Tanh)	-	1	0
Normalización	768	1	1536
Dropout	-	1	0
Lineal	768 x 512	1	393216
Normalización	512	1	1024
Dropout	-	1	0
Lineal	512 x 6	1	3072
Total	-	-	109,637,104

Cuadro 6.9: Resumen de las capas y parámetros del modelo BertRegressionModel

6.3.3. Entrenamiento

Para el entrenamiento se ha mantenido la proporción 0.9 para el conjunto de entrenamiento y 0.1 para la validación. Hay que destacar que se utiliza exactamente la misma

división en ambas arquitecturas para realizar una comparación justa.

Se entrena el modelo durante 50 épocas. Las primeras 10 mantienen las capas de transformadores congeladas y las 40 siguientes descongelan el modelo entero. Se utiliza el el error cuadrático medio como función de pérdida y el optimizador es AdamW, siguiendo la línea del modelo LSTM. Se utiliza una tasa de aprendizaje de $1 \cdot 10^{-2}$ para las primeras 10 épocas y $1 \cdot 10^{-4}$ para las últimas 40. A continuación se muestra una figura con la evolución de la función de pérdida para el entrenamiento tanto como para las categorías principales como para las subcategorías.

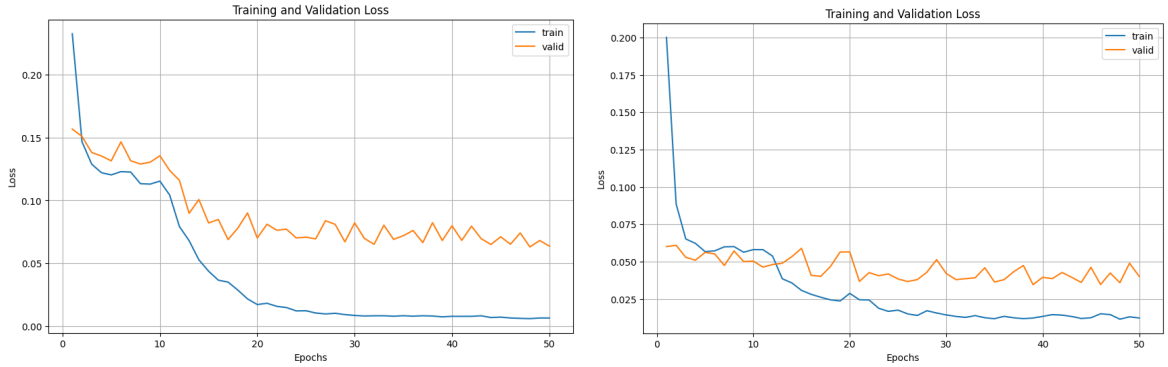


Figura 6.7: Evolución de la función de pérdida Bert perfiles (Base)

6.3.4. Validación perfiles

Tras el entrenamiento del modelo se reservaron exactamente las mismas observaciones para el conjunto test que para el modelo LSTM. A continuación en la tabla 6.10 se muestran los resultados para las métricas de evaluación usadas:

Métricas	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Media**
MSE	0.0669	0.1327	0.0440	0.0244	0.0551	0.0587	0.0636
MAE	0.1470	0.2256	0.0780	0.0556	0.1143	0.1094	0.1216
R2	0.6259	0.4143	0.5756	0.7767	0.3176	0.4129	0.5205

Cuadro 6.10: Métricas Evaluación (MSE, MAE, R2) Bert perfiles (Base)

- MSE: Para el MSE observamos valores en un rango entre 0,0244 y 0.1327. Con una media de 0,0636. Si lo comparamos con los valores de la tabla 6.3 obtenemos una media más baja pasando de 0,1042 a 0,0636. Además el modelo Bert obtiene una métrica más baja en todas las categorías excepto en 'P' y "L". Esto indica que en general, el modelo de Bert predice los valores más cercanos a los reales comparados con los LSTMs.

- MAE: Para el error medio absoluto obtenemos un rango de 0.0556 a 0.2256. Con una media de 0.1216, más baja que para el modelo LSTM. Esto nos indica que el modelo de Bert tiene una menor desviación promedio en las predicciones en comparación con el modelo LSTM.

- R2: En cuanto a la variabilidad explicada por el modelo, la media es de 0.52, superior a los 0.3791 de media del modelo LSTM. Si comparamos la categoría con menor variabilidad en el modelo Transformes es la categoría P con un R2 = 0.3176 y la categoría con mayor variabilidad se corresponde con L con un R2 0.7767. Si comparamos categoría a categoría los valores de R2 de la tabla 6.10 con la tabla 6.3 observamos que el modelo de Transformers presentan un valor de R2 más alto en todas las categorías excepto en "P", lo que indica que los Transformers tienen un mejor ajuste y, por lo tanto, explican mejor la variabilidad de los datos en comparación con los LSTM.

A continuación se va a aplicar el mismo enfoque visto en la sección anterior. En primer lugar se presenta en la tabla 6.11 la modelización numérica del modelo Transformers.

Perfil	**μ (error)**	**σ (error)**	**Corr. Coef. (R)**	**Std. Error of μ**
F	-0.025797	0.257300	0.795293	0.025352
E/C	0.047143	0.361176	0.673797	0.035588
A/T/A	-0.030290	0.207657	0.764636	0.020461
L	-0.002509	0.156111	0.881371	0.015382
P	-0.040369	0.231279	0.581778	0.022789
Ac	0.022192	0.241363	0.692400	0.023782

Cuadro 6.11: Validación modelo Bert perfiles (Base)

Observando la tabla 6.11 vemos que el error medio en todos los perfiles es cercano a 0, lo que nos indica predicciones precisas. La desviación estándar del error es baja en todos los perfiles, especialmente en perfiles como "L"(0.1561) y 'A/T/A' (0.2077), lo que sugiere una alta consistencia en las predicciones. El coeficiente de correlación (R) es alto en perfiles como "L"(0.8814) y "F"(0.7953), indicando una fuerte relación lineal entre las predicciones y los valores reales.

Si realizamos una comparativa con la tabla 6.4 observamos que el transformers presentan un error medio más cercano a cero y una desviación estándar más baja en todos los perfiles profesionales, indicando así predicciones más precisas y consistentes. Además, los transformers muestran un coeficiente de correlación (R) más alto, lo que sugiere una relación lineal más fuerte entre las predicciones y los valores reales. Finalmente, el error estándar también es menor en el modelo de transformers, lo que indica una mayor precisión en la estimación del error verdadero.

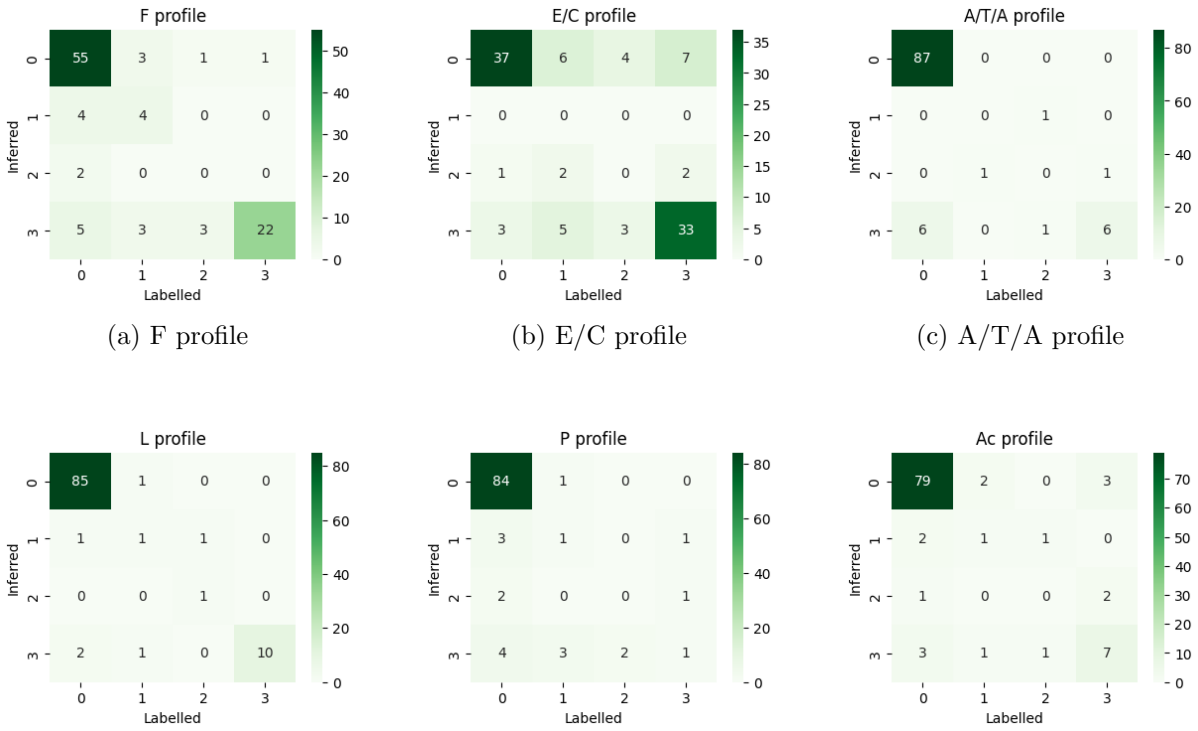
A continuación se muestra la tabla 6.12 donde se representa la tasa de aciertos del modelo cuando se produce la discretización de los valores en 4 categorías. El modelo de Bert presenta una precisión media del 83.17 %, un 5 % superior a la precisión obtenida con el modelo *AWD-LSTM*.

Perfil	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**
Test	78.641	67.961	90.291	94.175	83.495	84.466

Cuadro 6.12: Precisión modelo Bert perfil (Base)

Si comparamos los valores de esta tabla con la tabla 6.5 observamos que el modelo de transformers tiene una mayor tasa de acierto en cada una de los 6 perfiles. Podemos concluir que el modelo de transformers modela mejor los datos de las 6 categorías principales de los consejeros independientes tanto para una modelización numérica como discreta.

A continuación se muestran las matrices de confusión para los 6 diferentes perfiles para el conjunto test 6.8. Observamos que se sigue manteniendo la idea de una mayor representación de valores extremos.



(a) F profile (b) E/C profile (c) A/T/A profile

(d) L profile (e) P profile (f) Ac profile

Figura 6.8: Matrices de confusión perfiles modelo Bert

6.3.5. Validación subperfiles

Tras el entrenamiento del modelo para los 8 subperfiles se han calculado las métricas usadas para el resto de modelos base. A continuación en la tabla 6.13 se muestran los resultados obtenidos.

Metric	FB	FnB	CEO	E/C_F	E/C_M	E/C_RH	E/C_I	E/C_E	Media
MSE	0.02946	0.0062	0.02261	0.0957	0.0628	0.0486	0.06249	0.1047	0.0541
MAE	0.1037	0.0457	0.0682	0.12467	0.0926	0.06461	0.08754586	0.1517	0.0924
R2	0.6946	0.8939	0.73389	0.1052	0.12893	0.0781	0.124361	0.22797	0.3711

Cuadro 6.13: Métricas Evaluación (MSE, MAE, R2) Bert subperfiles (Base)

- Para la métrica MSE se obtiene una media de 0.0541. si la comparamos con el modelo equivalente con arquitectura LSTM vemos que se obtiene un valor algo

superior. Se observa una distribución bastante homogénea donde encontramos el mayor error cuadrático medio en la el subperfil E/C_E con un valor de 0.1047.

- Para la métrica MAE obtenemos un valor medio de 0.0924. Lo nos indica que la predicción total se desvía de la realidad un 9.24 % de media. Si lo comparamos con el modelo análogo con arquitectura AWD_LSTM se observa un valor superior de MAE para este modelo.

- En cuanto al R2, vemos que este modelo explica un 37 % de la variabilidad de los datos, un 5 % menos de variabilidad que el modelo LSTM. En cuento a los diferentes subperfiles vemos que las categorías con una mayor representación en el conjunto de datos como son FB,FnB y CEO explican una mayor variabilidad que en los subperifles menos representados E/C_F, E/C_M, E/C_RH, E/C_I, E/C_E.

Vamos a realizar ahora una evaluación en dos partes siguiendo la misma estructura que los anteriores modelos validados. En primer lugar vamos a evaluar como este modelo imita la forma de etiquetar humana. En la tabla 6.14 se pueden encontrar los resultados.

Subperfil	μ **(error)**	σ **(error)**	**corr. coef. (R)**	**std. error of** μ
FB	0.091957	0.144924	0.917836	0.015537
FnB	-0.000158	0.078769	0.952124	0.008445
CEO	0.006142	0.150247	0.866609	0.016108
E/C_F	-0.104888	0.290975	0.228953	0.031196
E/C_M	-0.085248	0.235684	0.062794	0.025268
E/C_RH	-0.059011	0.212503	0.049672	0.022783
E/C_I	-0.082033	0.236008	0.131020	0.025303
E/C_E	-0.140277	0.291571	0.068774	0.031260

Cuadro 6.14: Validación modelo Bert subperfiles (Base)

- En cuanto al error encontramos un error bastante cercano al 0. A excepción de los subperfiles menos representados (E/C_F,E/C_M, E/C_RH, E/C_I, E/C_E), que presentan valores cercanos a 0,1. Lo que indica una buena estimación entra las predicciones y los valores reales de los subperfiles FB, FnB y CEO. Vemos que los subperfiles FB y CEO tienen medias positivas, lo que indica que el modelo tiende a sobrestimar estos perfiles. Lo contrario pasa con el resto de subperfiles, estos son subestimados.

- Observando la columna del σ (error) vemos que los valores son bastante homogéneos. A excepción de FnB, que presenta un error σ bastante bajo. Lo que indica que la dispersión para este subperfil es relativamente baja, indicando así que el modelo es más más consistente en este perfil en comparación con el resto.

- Observando las correlaciones vemos valores muy extremos. Las correlaciones más altas concuerdan con los subperfiles más representados en el conjunto de datos. Lo que indica una buena correlación en las predicciones realizas por el modelo para esos subperfiles y los valores etiquetados. Todo lo contrario pasa para el resto de subperfiles.

- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

  A continuación se presenta la tabla 6.15 con las precisiones del modelo con los valores discretizados en 4 categorías de acuerdo a los intervalos definidos anteriormente. La media de precisión de este modelo para los subperfiles es del 85.61 % un 2 % inferior que la precisión obtenida con el modelo *AWD-LSTM*.

Subperfil	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**
Value	85.057	90.8046	85.057	83.91	85.057	90.805	87.356	75.862

Cuadro 6.15: Precisión modelo Bert subperfil (Base)

Tras el análisis podemos concluir que la que para la categorización de subperfiles sin ninguna técnica de aumento de datos el modelo *AWD-LSTM* categoriza mejor las biografías de los consejeros independientes en los 8 subperfiles definidos que el modelo *Bert*.

Capítulo 7

Implementación de técnicas de data augmentation

Durante este capítulo se explicará la implementación de las técnicas utilizadas de data augmentation en nuestro conjunto de datos.

A la vista de los resultados obtenidos en el capítulo anterior vemos que no han sido del todo satisfactorios, esto se debe en parte a la poca cantidad de datos etiquetados 1023.

Para abordar este problema, se ha decidido utilizar dos técnicas de aumento de datos (data augmentation). En este capítulo, se desarrollarán dos experimentos en los que se pretende observar la influencia del balacenceo del conjunto de entrenamiento y el *label smoothing* en los modelos base.

7.1. Conjunto de datos desbalanceado

El problema del desbalanceo de datos surge en muchas aplicaciones donde la clase positiva ocurre con poca frecuencia. Este desbalance puede ser de dos tipos diferentes: intrínseco, debido a la frecuencia natural de los datos, o extrínseco, causado por factores externos como procedimientos de recolección o almacenamiento.

Es crucial considerar la representación de las clases minoritarias y mayoritarias al realizar modelos de deep learning sobre datos desbalanceados. Se ha visto que se pueden obtener buenos resultados si ambos grupos están bien representados y provienen de distribuciones no superpuestas. Al examinarse los efectos del desbalance, creando conjuntos de datos artificiales con diferentes combinaciones de complejidad y tamaño de entrenamiento, se encontró que la sensibilidad al desbalance aumenta con la complejidad del problema.

En nuestro campo de biografías de consejeros independientes, debido a la escasez de datos etiquetados y la escasez de representación de algunos perfiles y subperfiles, hace que aprender de datos extremadamente desbalanceados, donde la clase minoritaria representa solo el 0.1 % de los datos de entrenamiento, suponga un reto. Este desbalanceo de las categorías es algo negativo a la hora del entrenamiento de nuestros modelos, ya que perjudican a las clases minoritarias [22].

El desbalanceo se puede tratar de diversas formas. A continuación se muestran varias [34]:

- **Ajuste de Parámetros del modelo**: Consiste en ajustar parámetros ó métricas del propio algoritmo para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento.
- **Modificar el Dataset:** Podemos eliminar muestras de la clase mayoritaria para reducirlo e intentar equilibrar la situación.
- **Muestras artificiales:** Podemos intentar crear muestras sintéticas (no idénticas) utilizando diversos algoritmos que intentan seguir la tendencia del grupo minoritario. Según el método, podemos mejorar los resultados. Lo peligroso de crear muestras sintéticas es que podemos alterar la distribución "natural" de esa clase y confundir al modelo en su clasificación.
- **Métodos de Ensamble Balanceado** (Balanced Ensemble Methods): Es una técnica donde se combinan múltiples modelos de aprendizaje, éstos son ajustados para dar igual importancia a todas las clases, mejorando así la precisión en la clasificación de clases minoritarias. Utilizan técnicas como el submuestreo de la clase mayoritaria o el sobremuestreo de la clase minoritaria para lograr un conjunto de datos más equilibrado antes de entrenar los modelos de ensamble.

7.2. Smooth Labeling

Como se ha mencionado en el marco teórico, Label smoothing es una técnica utilizada para regularizar los modelos de clasificación. Consiste en suavizar las etiquetas de las clases, en lugar de asignar un valor estricto, como ha realizado nuestro experto humano al etiquetar biografías. La técnica se basa en el aplanado de las diferentes etiquetas dadas asignado una distribución uniforme a todas las clases. Esta técnica ayuda a los modelos a ser menos confiados en sus predicciones y, por lo tanto, mejora su capacidad de generalización. En la figura 7.1 se puede observar un ejemplo de la aplicación de la técnica label smoothing para 9 clases.

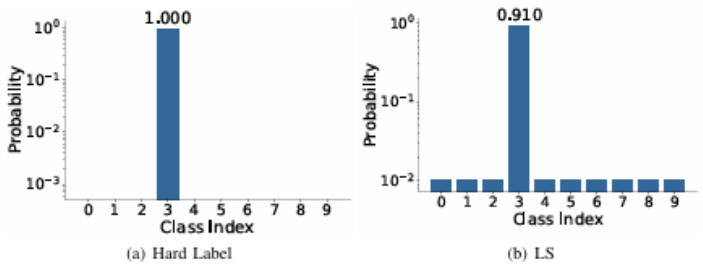


Figura 7.1: Ejemplo Label Smoothing

7.3. Experimentación

7.3.1. Diseño

El diseño de los experimentos a realizar va a constar de dos fases: Primera es la realización de un experimento base que servirá como punto inicial para la aplicación de las diferentes modificaciones realizadas. En la segunda fase se realizaran dos experimentos. El primer experimento utilizara *label smoothing* sobre el conjunto de datos de entrenamiento. En el otro experimento se aplicará un balanceo al conjunto de datos mediante la introducción de muestras artificiales. Posteriormente se compararán los resultados con el experimento base.

El experimento base se realizó en el **Capítulo 6: Modelos de Regresión**. tanto para perfiles como subperfiles para los modelos de arquitectura AWD-LSTM y Bert.

Durante la próxima sección se redactarán los dos experimentos aplicando la técnica de *label smoothing* directamente sin tener en cuenta el desbalanceo en el dataset inicial y por otra parte se realiza el balanceo del dataset mediante la introducción de un pequeño ruido.

7.3.2. Experimento 1

Dada la poca cantidad de datos etiquetados para poder obtener resultados realmente buenos para este problema complejo de clasificación de las biografías de los consejeros independientes de perfiles y subperfiles. Se propone la utilización de la técnica de *label smoothing* para ampliar el espacio de características y así disponer de más datos de entrenamiento. A continuación se muestra la implementación de esta técnica.

```
1
2  def smooth_labeling(df):
3
4    ruido = 0.025
5    new_rows = []
6    for i in range(df.shape[0]):
7
8      new_row = df.iloc[i].copy()
9      for j in columnas_y:
10
11       if(random.randint(1,3) == 1):
12         if(new_row[j] != 1):
13           new_row[j] += 0.025
14
15       elif(random.randint(1,3) == 2):
16         if(new_row[j] != 0):
17           new_row[j] -= 0.025
18
19     new_rows.append(new_row)
20
21   return pd.DataFrame(new_rows)
```

Listing 7.1: Código Implementación Smooth Labeling

Esta función suma o resta un ruido de ±0,025 a los valores de los diferentes perfiles. Para cada fila del conjunto de datos, selecciona aleatoriamente si debe aumentar, disminuir o dejar sin ningún cambio el valor de cada perfil. El resultado es un nuevo conjunto de datos con las etiquetas suavizadas. Con la obtención de este nuevo conjunto de datos se añade al conjunto de datos de entrenamiento original duplicando así el tamaño del

conjunto original.

Cabe destacar que los modelos AWD-LSTM y Bert se han mantenido constantes con respecto a los modelos base. La creación de los conjuntos de datos de prueba es igual que en los modelos base tanto para perfiles como subperfiles. Con esto se busca una comparación lo más objetiva posible.

Experimento 1: AWD-LSTM perfiles

Para la realización de este experimento se ha entrenado el modelo LSTM para categorías principales con un total de 1840 biografías. La misma distribución de este conjunto de entrenamiento ampliado es el mismo que el conjunto de datos base. Se ha entrenado para un total de 250 épocas con una función de pérdida MSE (Error Cuadrático Medio) y un optimizador Adam. En la figura 7.2 se presenta la evolución de la función de pérdida para el conjunto de entrenamiento y validación utilizados. Se puede observar que el valor de esta función se estabiliza en un valor más bajo comparándolos con la figura 6.4.

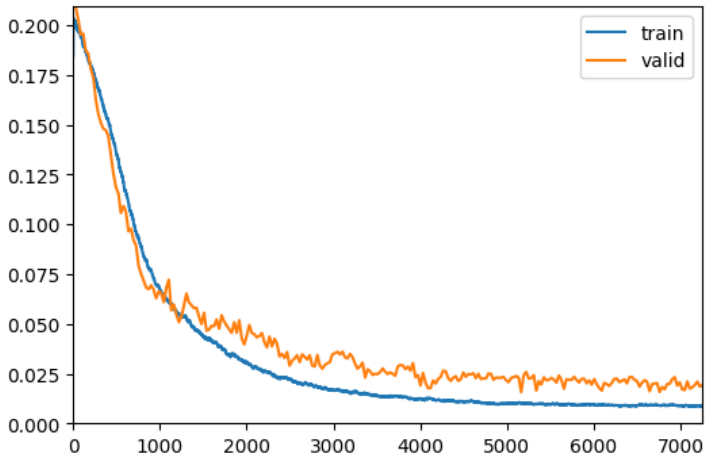


Figura 7.2: Evolución de la función de pérdida modelo AWD-LSTM perfiles (Experimento 1)

A continuación se presenta la evaluación del modelo para las métricas utilizadas (MSE, MAE y R2). Estas se pueden encontrar en la tabla 7.1.

Métricas	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Media**
MSE	0.1126	0.2059	0.0678	0.1393	0.0393	0.0570	0.1036
MAE	0.1907	0.2819	0.0935	0.1781	0.0881	0.1032	0.1559
R2	0.3701	0.0912	0.3470	0.2758	0.5135	0.4302	0.3379

Cuadro 7.1: Métricas Evaluación (MSE, MAE, R2) AWD-LSTM perfiles (Experimento 1)

- MSE: Observando el valor obtenido con la métrica de MSE muestra un valor promedio de 0.1036, con la categoría E/C presentando el mayor error (0.2059) y la categoría P el menor (0.0393), al igual que el modelo base. Comparando las medias del modelo base y el experimento 1 observamos que se obtiene un error ligeramente superior con el modelo base.

- MAE: Observamos un valor medio del MAE de 0.1559. Comparando la media con el modelo base obtenemos que se produce un error menor con la utilización del *label smoothing*. Lo que indica que el modelo con el experimento 1 se desvía menos de la realidad que el modelo base.

- R2: La métrica R2 presenta un valor medio de 0.3379, sugiere que el modelo explica una proporción moderada de la variabilidad en los datos. La categoría P tiene el valor más alto (0.5135), indicando una mejor capacidad del modelo para explicar la varianza en esta categoría. Por otro lado, E/C tiene el valor más bajo (0.0912), lo que sugiere que el modelo no captura bien la variabilidad en esta categoría. Si comparamos las métricas con el modelo base se observa que la variabilidad explicada a disminuido alrededor de un 4 %.

A raíz de estos resultados se realiza una comparativa con el modelo LSTM base para categorías principales para ver si se ha producido alguna mejora. Se presenta en la figura 7.3 la comparativa para estas métricas.

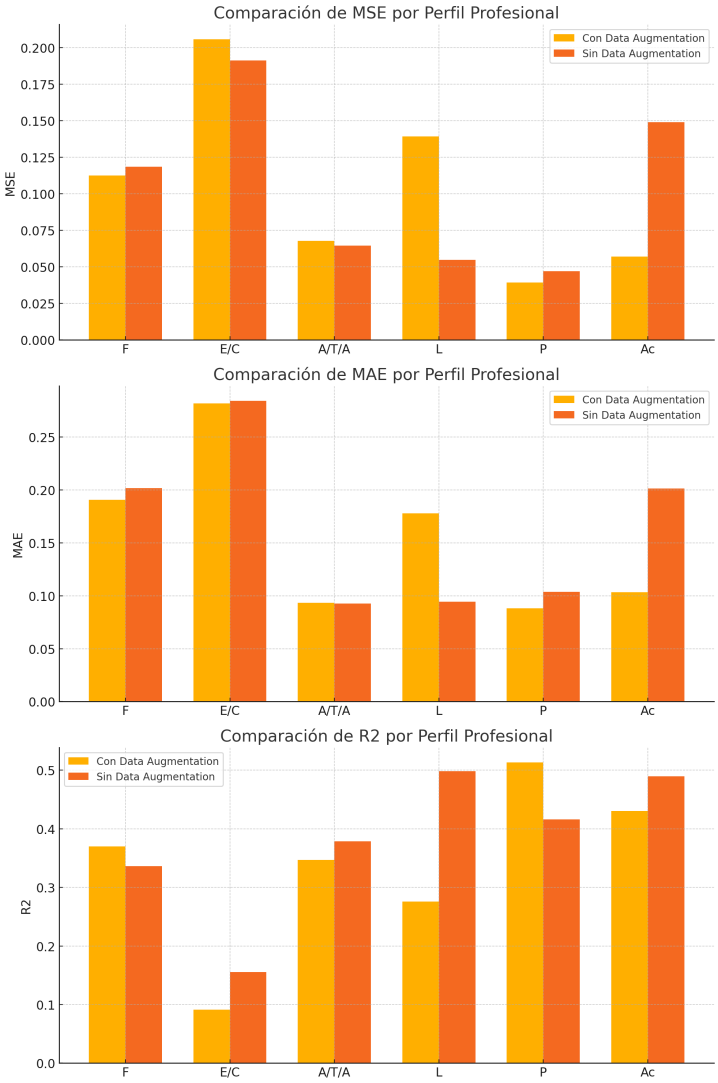


Figura 7.3: Comparación de métricas modelos AWD-LSTM perfiles (Base y Experimento 1)

Se realiza ahora la validación en dos partes como se ha realizado en los modelos base. En la tabla 7.2 de muestran los resultados numéricos para lo modelización tratando de imitar el etiquetado humano.

Perfil	**μ (error)**	**σ (error)**	**corr. coef. (R)**	**std. error of μ**
F	-0.035285	0.331656	0.646976	0.032679
E/C	0.032296	0.430166	0.534478	0.042386
A/T/A	-0.054286	0.257551	0.604754	0.025377
L	-0.027790	0.244435	0.678212	0.024085
P	-0.033596	0.186566	0.759485	0.018383
Ac	0.070352	0.381079	0.339858	0.037549

Cuadro 7.2: Validación modelo AWD-LSTM perfiles (Experimento 1)

Observando la tabla el error medio para los diferentes perfiles observamos valores bastante centrados entorno al 0, a excepción del perfil Ac que presenta un valor próximo a $0,1$. Observamos que los perfiles E/C y Ac presentan medias positivas. Esto indica que el modelo sobrestima las predicciones para estos perfiles. Lo contrario ocurre para los perfiles con medias negativas.

Si observamos lo valores de la desviación estándar del error varía desde 0.186566 para el perfil P, hasta 0.430166 para el perfil E/C. Esto indica que el perfil P tiene la variabilidad más baja en sus errores de predicción, lo que sugiere una mayor consistencia del modelo para este perfil. En contraste, la alta desviación estándar en E/C indica una mayor dispersión de los errores y, por lo tanto, menor consistencia en las predicciones para este perfil.

El coeficiente de correlación varía entre 0.339858 para el perfil Ac, indicando una relación más débil entre las predicciones del modelo y los valores reales, y 0.759485 para el perfil P, que muestra una relación fuerte. En general, el modelo tiene una correlación aceptable con los datos reales.

Finalmente el error estándar del error medio es más bajo. Esto indica que el modelo en general tiene una precisión aceptable en la estimación del error medio.

Realizamos ahora la discretización de las valores en 4 categorías para los siguientes intervalos, tal y como se ha realizado para los modelos base. A continuación se muestra una tabla con la precisión obtenida para este experimento y la figura que compara la precisión del modelo base y el experimento 1 para el modelo LSTM en las seis categorías principales.

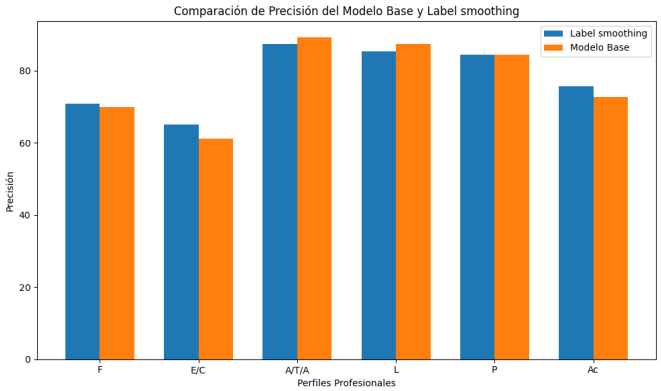


Figura 7.4: Comparación precisión modelos AWD-LSTM perfiles (Base y Experimento 1)

Perfil	Precisión
F	72.815534
E/C	70.873786
A/T/A	89.320388
L	92.233010
P	85.436893
Ac	82.524272

Figura 7.5: Precisión modelo AWD-LSTM perfiles (Experimento 1)

Obtenemos una precisión media del 82,2 %, aproximadamente un 5 % superior que el modelo base. En particular, los perfiles F y E/C muestran mejoras notables, pasando de valores cercanos al 69.9 % y 61.2 % a aproximadamente 70.9 % y 65.0 %, respectivamente. Por otro lado , los perfiles A/T/A, L, y P mantiene porcentajes de precisión similares. Esto sugiere que el uso de la técnica *label smoothing* ha mejorado en rasgos generales la precisión del modelo LSTM para los seis diferentes perfiles.

Experimento 1: AWD-LSTM subperfiles

Para la realización de este experimento se ha entrenado el modelo LSTM para los 8 subperfiles con un total de 1840 biografías. Este conjunto de datos sigue la misma distribución que el conjunto de datos base.

Se ha entrenado para un total de 250 épocas con una función de pérdida MSE (Error Cuadrático Medio) y un optimizador Adam. En la figura 7.6 se presenta evolución de la función de pérdida para el conjunto de entrenamiento y validación utilizado para el modelo LSTM de subperfiles.

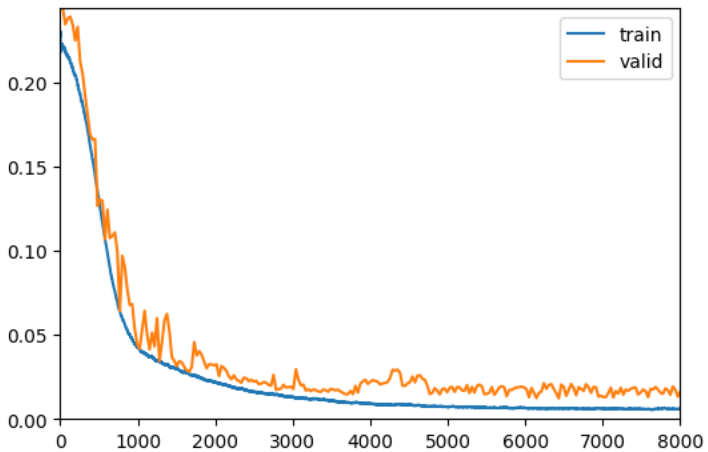


Figura 7.6: Evolución de la función de pérdida modelo AWD-LSTM subperfiles (Experimento 1)

En la tabla 7.3 se presenta las métricas (MSE, MAE y R2) para la evaluación del modelo para el conjunto test.

Metric	FB	FnB	CEO	E/C_F	E/C_M	E/C_RH	E/C_I	E/C_E	Media
MSE	0.0044	0.0074	0.0337	0.0065	0.0499	0.0489	0.0108	0.0238	0.0232
MAE	0.0307	0.0417	0.0783	0.0288	0.0919	0.0646	0.0416	0.0667	0.0555
R2	0.9548	0.8738	0.6034	0.9245	0.1036	0.0850	0.8050	0.7212	0.6127

Cuadro 7.3: Métricas Evaluación (MSE, MAE, R2) AWD-LSTM subperfiles (Experimento 1)

- El MSE presenta un media de 0.0232, lo que indica que el error cuadrático es bajo. En comparación con el modelo base se mejora la métrica en un 2 % aproximadamente pasando de un 0.04 a un 0.023. Lo que indica una mejora general del modelo con *label smoothing*.
- El MAE presenta una media de 0.0555, lo que significa que, las predicciones del modelo están desviadas en aproximadamente 0.0555 unidades respecto a los valores reales. En comparación con el modelo base supone una mejora en la desviación de las predicciones de aproximadamente un 2 %.
- El R2 presenta una media 0.6127, indicando que aproximadamente el 61.27 % de la variabilidad en los datos es explicada por el modelo. Observamos como los perfiles FB y FnB, explican una variabilidad próxima a 1, otros perfiles como E/C_RH y E/C_M, presentan una variabilidad explicada muy baja lo que indica que el modelo no es igualmente efectivo en todos los subperfiles. En comparación con el r2 del modelo base encontramos una mejora sustancial, se ha pasado de una variabilidad media del 41.44 % al 61.27 %.

A continuación se presenta una comparativa del modelo LSTM base para los 8 subperfiles con el modelo del experimento 1.

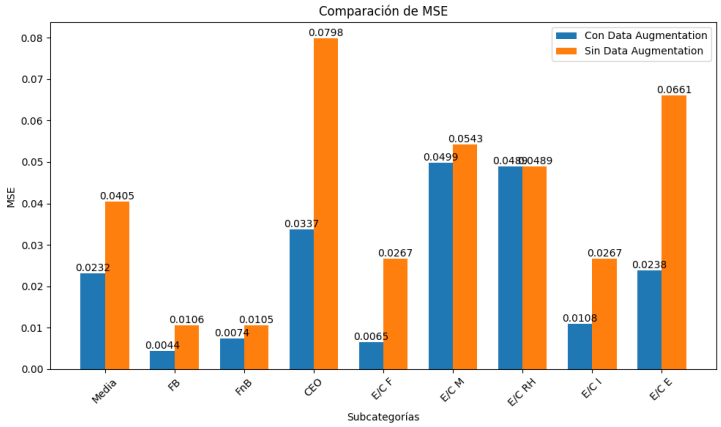


Figura 7.7: Comparativa MSE modelo AWD-LSTM subperfiles (Base, Experimento 1)

A continuación se realiza la evaluación del modelo del experimento 1 para lo modelización tratando de imitar el etiquetado humano. En la tabla 7.4 se muestran los resultados obtenidos.

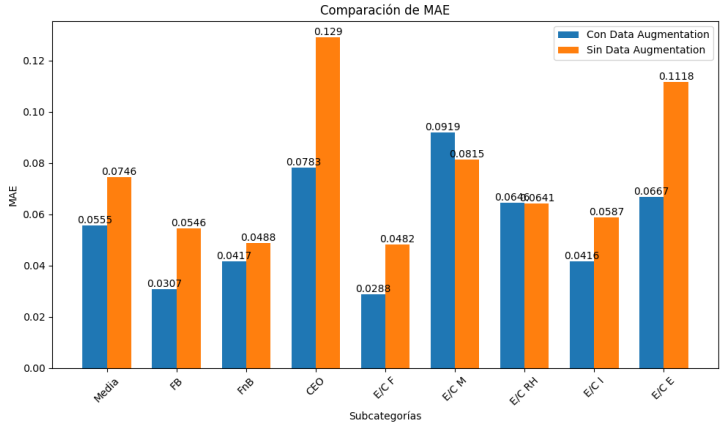


Figura 7.8: Comparativa MAE modelo AWD-LSTM subperfiles (Base, Experimento 1)

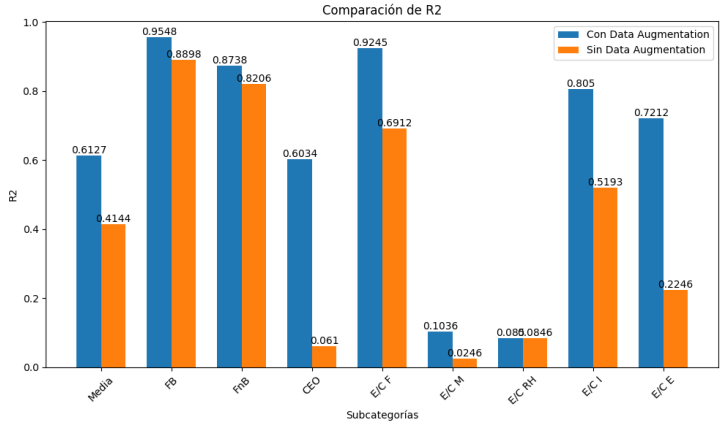


Figura 7.9: Comparativa R2 modelo AWD-LSTM subperfiles (Base, Experimento 1)

Subperfil	μ **(error)**	σ **(error)**	**Corr. Coef. (R)**	**Std. Error of** μ
FB	0.019252	0.063137	0.981557	0.006769
FnB	0.021171	0.083292	0.947626	0.008930
CEO	0.043386	0.178387	0.825481	0.019125
E/C_F	0.004968	0.080672	0.963869	0.008649
E/C_M	0.064441	0.213831	0.695588	0.022925
E/C_RH	-0.059472	0.213102	0.157041	0.022847
E/C_I	0.009357	0.103631	0.901231	0.011110
E/C_E	-0.017474	0.153186	0.856852	0.016423

Cuadro 7.4: Validación modelo AWD-LSTM subperfiles (Experimento 1)

- El error medio (μ) es más bajo para los subperfiles E/C_F y FB, esto indica indica una mayor precisión del modelo en estas categorías. Por otro lado, el subperfil E/C_RH presenta un error negativo, lo que sugiere una subestimación en las predicciones del modelo para este subperfil, lo contrario ocurre para los subperfiles con errores medios positivos.

- En cuanto al error (σ) los subperfiles E/C_M y E/C_RH tienen el mayor error. Esto indica una variabilidad significativa en sus predicciones. Sin embargo, las categorías con la menor dispersión son FB y E/C, lo que indica una mayor coherencia en las

predicciones para estos subperfiles.

- Los subperfiles FB y E/C_F tuvieron coeficientes de correlación muy cercanos a 1. Lo que indica un buen ajuste del modelo para ellos. Sin embargo, el subperfil E/C_RH tiene un coeficiente de correlación muy bajo, lo que indica que el modelo no captura adecuadamente las patrones de este subperfil.
- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

A continuación se produce la disctretización de los valores en cuatro categorías. En la tabla 7.11 se pueden encontrar las precisiones de este modelo. En la figura 7.10 se observa un comparativa

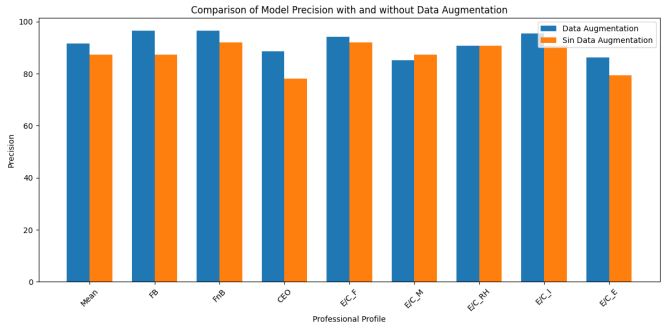


Figura 7.10: Comparación precisión modelos LSTM para subperfiles (Base y Experimento 1)

Perfil	Precisión
FB	96.551724
FnB	96.551724
CEO	88.505747
E/C_F	94.252874
E/C_M	85.057471
E/C_RH	90.804598
E/C_I	95.402299
E/C_E	86.206897

Figura 7.11: Precisión modelo AWD-LSTM subperfiles (Experimento 1)

Observando el gráfico, podemos ver que la precisión media del modelo ha aumentado de 87.21 % a 91.67 %. Al analizar los subperfiles individualmente, notamos que la precisión ha mejorado en la mayoría de las categorías, alcanzando una mejora máxima del 10 % en la categoría de CEO. Esto indica que la técnica de data augmentation mejora significativamente el rendimiento del modelo AWD-LSTM para la clasificación de biografías en los diferentes subperfiles.

Experimento 1: Bert perfiles

Para la realización de este experimento se ha mantenido la misma estructura que el modelo base *Bert*. Se ha entrenado para un total de 50 épocas, siguiendo la misma estructura de congelamiento y descongelamiento de las capas. Durante las primeras 10 épocas se ha entrenado el modelo con las capas congeladas con un *learning rate* de $1 \cdot 10^{-2}$. Para las 40 épocas restantes se han descongelado las capas y se ha utilizado un *learning rate* de $1 \cdot 10^{-4}$. Como función de pérdida se ha utilizado el error cuadrático medio (MSE) y un optimizador Adam.

A continuación se muestra la figura 7.12 con la evolución de la función de pérdida del modelo para el conjunto de entrenamiento y validación.

A continuación se realiza la evaluación del modelo para las métricas (MSE, MAE y R2). Los resultados obtenidos se muestran en la siguiente tabla.

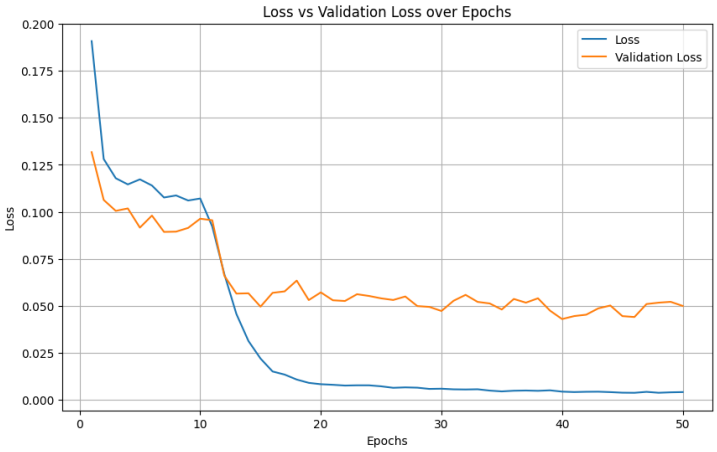


Figura 7.12: Evolución de la función de pérdida Bert experimento 1 (Perfiles)

Métricas	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Media**
MSE	0.0676	0.1724	0.0487	0.0275	0.0463	0.0601	0.0704
MAE	0.1561	0.2569	0.0927	0.0536	0.0963	0.1131	0.1281
R2	0.6218	0.2389	0.5306	0.7484	0.4265	0.3992	0.4942

Cuadro 7.5: Métricas Evaluación (MSE, MAE, R2) Bert perfiles (Experimento 1)

- MSE: Vemos que el valor medio es 0.0704 algo más alto que en el modelo base. Los diferentes perfiles presentan valores bastante similares salvo el perfil E/C donde tiene un MSE relativamente alto de 0.1724.
- MAE: Vemos que presenta un error medio absoluto de 12.81 algo más alto que en el modelo base. Lo que nos indica que la predicción total se desvía algo más para el modelo de Bert utilizando la técnica de *label smoothing* que sin ella.
- R2: El modelo de Bert para las categorías principales explica un 49.42 % de la variabilidad de los datos, algo más bajo que el modelo base. Destaca el valor de R2 en el perfil E/C donde encontramos una bajada de alrededor del 20 %. Aunque se han visto una mejora en la variabilidad en el perfil P, uno que tenía una menor representación en el conjunto de datos inicial. Esto nos indica que aumentar únicamente el espacio de características manteniendo la proporción de los perfiles no nos garantiza una mejora con respecto a explicar una mayor variabilidad en los datos.

A continuación se muestra la figura 7.13 donde se muestra una comparativa de las métricas donde se pueden apreciar mejor estas diferencias.

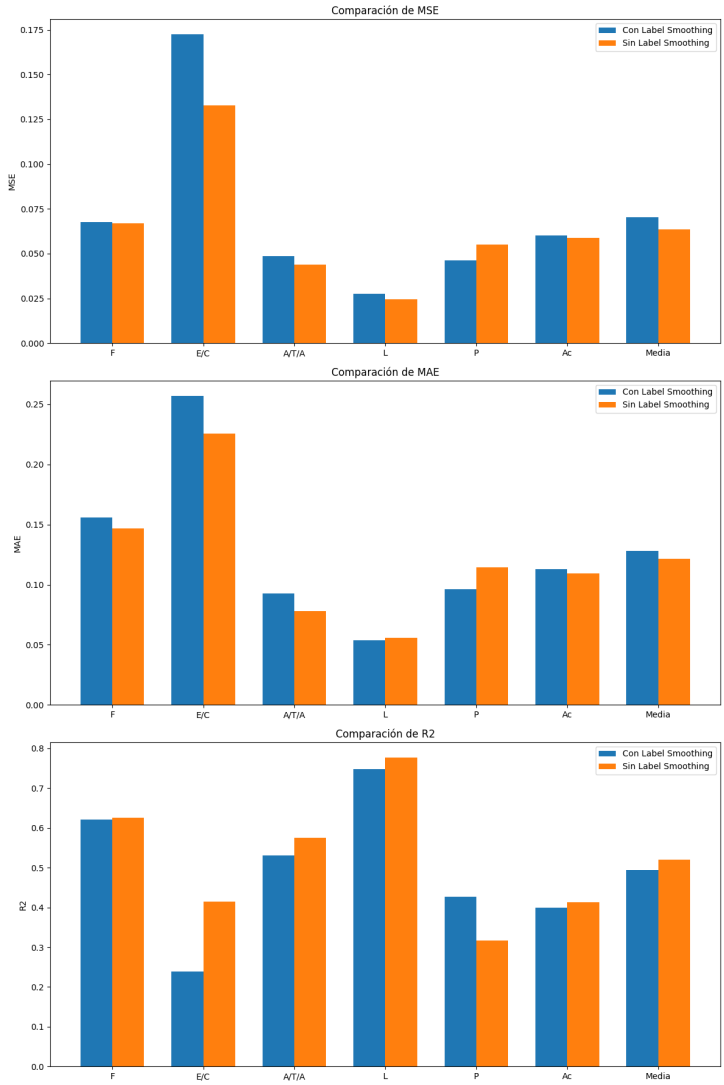


Figura 7.13: Comparativa métricas (MSE, MAE y R2) Experimento 1 con *Bert* (Perfiles)

Tras la evaluación del modelo vamos a realizar la validación que llevamos haciendo para todos los modelos. Primero se realiza la validación numérica donde se muestra la capacidad del modelo para imitar el etiquetado humano. En la tabla 7.6 se muestran los resultados para los seis perfiles principales para el modelo de *Bert*.

Perfil	μ (error)	σ (error)	corr. coef. (R)	std. error of μ
F	0.020226	0.259211	0.793981	0.025541
E/C	0.182693	0.372872	0.648057	0.036740
A/T/A	-0.015850	0.220122	0.733382	0.021689
L	-0.018333	0.164700	0.866910	0.016228
P	-0.041572	0.211178	0.669963	0.020808
Ac	0.017543	0.244565	0.679707	0.024098

Cuadro 7.6: Validación numérica del Modelo Bert Experimento 1 (Perfiles)

A continuación se presentan una tabla de precisión del modelo con los datos discretizados 7.7 y una figura comparativa con el modelo base análogo 7.14.

Perfil	Precisión
F	72.815534
E/C	70.873786
A/T/A	89.320388
L	92.233010
P	85.436893
Ac	82.524272

Cuadro 7.7: Precisión del modelo Bert Experimento 1 (Perfiles)

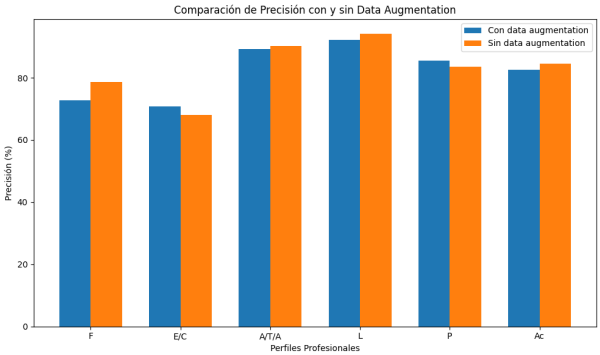


Figura 7.14: Comparativa de precisión del modelo Bert (perfiles) modelo base, experimento 1

Observando la tabla de precisión del modelo con el uso de *label smoothing* observamos una precisión media del 82.2 % muy similar al modelo base. Las precisiones son bastante buenas en todos los perfiles, superando una precisión del 70 % en cada uno de ellos. Llegando a tener una precisión máxima para el perfil L del 92 %. Si observamos la figura vemos que no hay una mejora general en la precisión con respecto al modelo sin aplicar el *label smoothing*.

Experimento 1: Bert subperfiles

Para la realización de este experimento se aplica la técnica de *label smoothing* para el modelo *Bert* para los subperfiles. Sigue la misma estructura que el modelo para los perfiles. A continuación se presenta la evolución de la función de pérdida durante las 50 épocas para el conjunto de entrenamiento y validación utilizados 7.15.

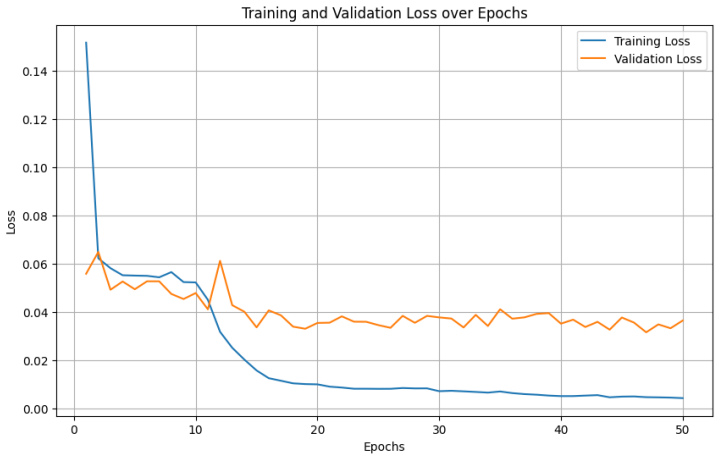


Figura 7.15: Evolución de la función de pérdida para el modelo Bert Experimento 1 (Subperfiles)

A continuación se realiza la evaluación con las métricas que venimos utilizando para los modelos base y el experimento 1. En la tabla 7.8 se muestran los resultados obtenidos.

Métricas	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**	**Media**
MSE	0.0132	0.0034	0.0263	0.0199	0.0356	0.0465	0.0613	0.0356	0.0307
MAE	0.0561	0.0316	0.0831	0.0723	0.0739	0.0743	0.0917	0.0759	0.0699
R2	0.8633	0.9424	0.6909	0.7697	0.3605	0.0310	0.1034	0.5824	0.5429

Cuadro 7.8: Métricas Evaluación (MSE, MAE, R2) Bert subperfiles (Experimento 1)

- Para la métrica MSE observamos un valor medio de 0.0307. En comparación con el valor obtenido con el modelo base, supone una mejora del 2 %. Esto indica que el modelo aplicando *label smoothing* realiza unas predicciones más precisas que el modelo base.
- Observando la métrica MAE podemos ver valores bastante homogéneos, con un MAE medio de 0.0699. En comparación con el valor obtenido con el modelo base, supone una mejora aproximada del 2 %. Esto indica que el modelo con *label smoothing* se desvía un 2 % menos de los valores reales que el modelo base.
- En cuanto al R2, observamos un valor medio de 0.5429, con el subperfil FnB con la variabilidad explicada más alta y el subperfil E/C_RH con la variabilidad explicada más baja. Si comparamos el R2 medio con el modelo base observamos que el modelo con *label smoothing* explica un 15 % más de variabilidad que el modelo base.

  A continuación se presentan tres gráficos comparativos entre el modelo base y el modelo del experimento 1, para las métricas analizadas anteriormente.

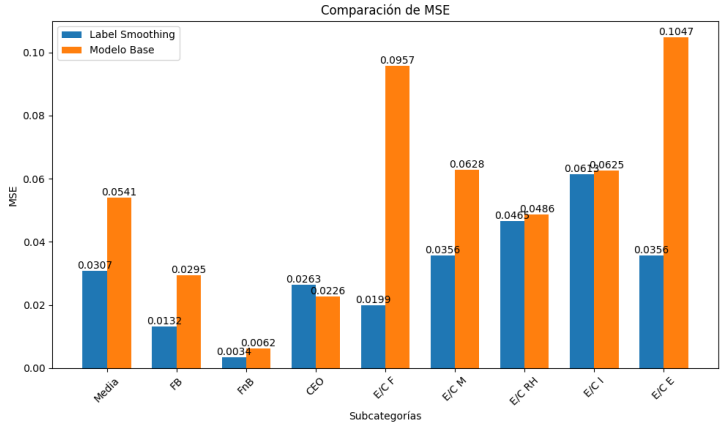


Figura 7.16: Comparativa MSE modelo base vs Bert experimento 1 (subperfiles)

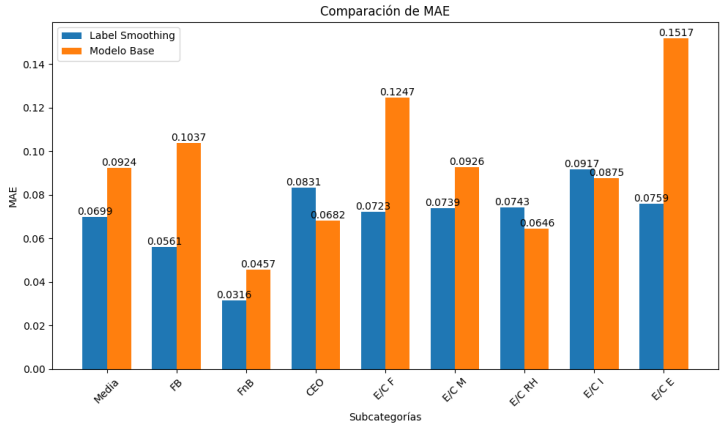


Figura 7.17: Comparativa MAE modelo base vs Bert experimento 1 (subperfiles)

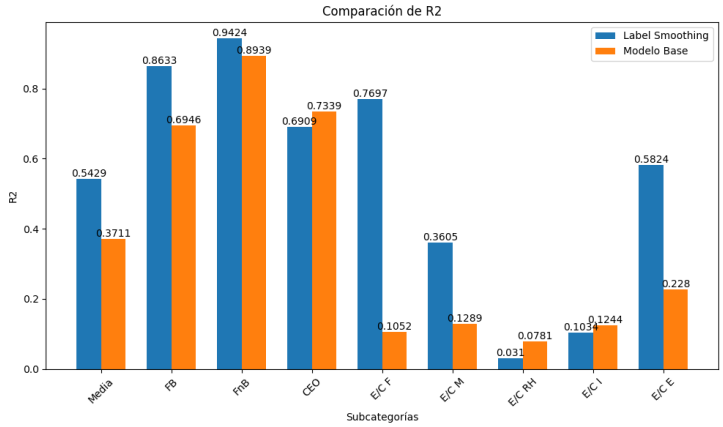


Figura 7.18: Comparativa R2 modelo base vs Bert experimento 1 (subperfiles)

Se realiza la validación en dos pasos como hemos venido haciendo hasta ahora. Se presenta en la tabla 7.9 los resultados obtenidos al realizar la validación del modelo para ver como replica el etiquetado humano.

- Observamos que el error medio es bastante próximo a cero en la mayoría de los subperfiles, a excepción del subperfil E/C_I que presenta un error medio más cercano al 0.1. Observamos que algunos subperfiles como FB y CEO presentan medias

Subperfil	μ **(error)**	σ **(error)**	**corr. coef. (R)**	**std. error of** μ
FB	0.036163	0.108995	0.945878	0.011686
FnB	0.007671	0.057549	0.971295	0.006170
CEO	0.043712	0.156067	0.862660	0.016732
E/C_F	0.014752	0.140423	0.879517	0.015055
E/C_M	0.008828	0.188420	0.647583	0.020201
E/C_RH	-0.045115	0.210894	0.131939	0.022610
E/C_I	-0.076421	0.235425	0.043971	0.025240
E/C_E	-0.019353	0.187690	0.769926	0.020122

Cuadro 7.9: Validación modelo Bert subperfiles (Experimento 1)

positivas, esto indica que el modelo tiende a sobrestimar el modelo en estos subperfiles. Lo contrario pasa en algunos subperfiles como E/C_RH o E/C_E. En estos subperfiles con media negativa se tiende a subestimar las predicciones.

- En cuanto al error σ vemos que los valores son bastante homogéneos, a excepción del subperfil FnB, que presenta un error σ 0,057. Esto indica que la dispersión del modelo para este subperfil es bastante baja en comparación al resto de subperfiles.

- Observando las correlaciones de los subperfiles encontramos correlaciones muy altas para perfiles como FB, FnB y CEO (subperfiles más representados), algunas correlaciones con valores medios, como los subperfiles E/C_M y E/C_E y correlaciones muy bajas como los subperfiles E/C_RH, E/C_I (subperfiles menos representados). Esto indica que el modelo tiende a correlacionar con mayor fuerza las predicciones con los valores reales para los subperfiles con mayor representación en el conjunto de datos.

- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

A continuación, se realiza la discretización de los valores en 4 categorías de acuerdo a los intervalos definidos anteriormente. En la tabla 7.10 se presenta la precisión de los subperfiles para el modelo *Bert* con la técnica de *label smoothing* aplicada. Se obtiene una precisión media del 88.936 %, lo que indica una mejora del 3 % aproximada con respecto al modelo base. En la figura 7.19 se obtiene una comparativa de la precisión de los subperfiles para el modelo base y el experimento 1.

Podemos concluir, que el modelo *Bert* para los subperfiles ha presentado mejoras sustanciales con la aplicación de la técnica *label smoothing*.

Perfil	Precisión
FB	91.954
FnB	94.253
CEO	83.908
E/C_F	88.506
E/C_M	89.655
E/C_RH	90.805
E/C_I	87.356
E/C_E	85.057

Cuadro 7.10: Precisión del modelo Bert Experimento 1 (Subperfiles)

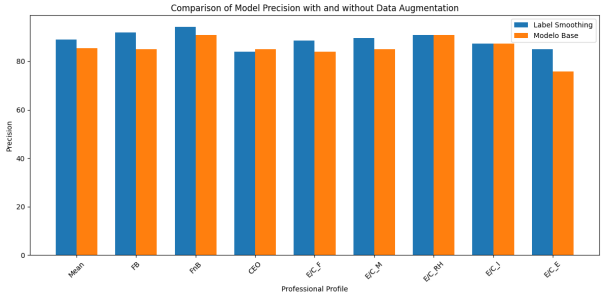


Figura 7.19: Comparativa de precisión del modelo Bert (subperfiles) modelo base, experimento 1

7.3.3. Experimento 2

Dada la naturaleza de nuestros datos se ha decidido aplicar la inclusión de muestras artificiales. Para ello nos hemos basado en la técnica SMOTE (Synthetic Minority Oversampling Technique)[9]. Esta técnica es un método de sobremuestreo sintético que genera nuevas muestras de la clase minoritaria mediante la interpolación entre una muestra de la clase minoritaria y sus vecinos más cercanos, con el objetivo de equilibrar la distribución de clases en el conjunto de datos. Por ello se han obtenido un subconjunto del dataset original, donde las filas seleccionadas correspondientes a los perfiles y subperfiles más representados, tienen un valor de 0. Para los perfiles las categorías más representadas son (F y E/C) y para los subperfiles son (FB, FnB y CEO).

A continuación, se ha aplicado la función *df_balancing*. Esta función está diseñada para aumentar las filas del conjunto de datos df en categorías específicas que están subrepresentadas, según los límites definidos por el umbral declarado en [22]. Tras la aplicación de esta función sobre el conjunto de datos se ha concatenado con el conjunto de datos original. De esta forma obtenemos un dataset más equilibrado. A continuación se define la función *df_balancing* para el balanceo de los subperfiles.

```
1
2  def df_balancing(df, columnas_y):
3      # Obtener la frecuencia de cada perfil principal
4      category_counts = [0, 0, 0, 0, 0, 0,0,0]
5      max_count = [0, 50, 0, 200, 200, 175, 200, 250]
6      new_rows = []
7      seen_rows = set()  # Un set para almacenar representaciones unicas de las filas
8
9      # Generar nuevas observaciones para categorias menos representadas
10
11     stop = False
12     while stop == False:
13       contador = 0
14       for i, cont in enumerate(columnas_y):
15         if(category_counts[i] < max_count[i]):
16           contador = contador + 1
17
18       print(category_counts)
19       if(contador == 0):
20         stop = True
21         new_rows_df = pd.DataFrame(new_rows)
22         return new_rows_df
23
24
25       new_row = df.sample(n=1).iloc[0].copy()
```

```
      count = 0
      for i, valor in enumerate(columnas_y):
        if new_row[valor] > 0 and (category_counts[i] + 1 > max_count[i]):
          count += 1

    # Si no hay conflictos, actualizar los valores y agregar la nueva fila
      if count == 0:

        for i, cont in enumerate(columnas_y):

          if(category_counts[i] < max_count[i]):

            if(random.randint(1,3) == 1):
              if(new_row[cont] != 1):
                new_row[cont] += 0.025

            elif(random.randint(1,3) == 2):
              if(new_row[cont] != 0):
                new_row[cont] -= 0.025

        row_hash = hash(tuple(new_row.values))  # Crear un hash de la fila
        if row_hash not in seen_rows:
          new_rows.append(new_row)
          seen_rows.add(row_hash)  # Anadir el hash de la nueva fila al set de control
          for i, nombre in enumerate(columnas_y):
            if new_row[nombre] > 0:
              category_counts[i] += 1
```

Experimento 2: AWD-LSTM perfiles

Para la realización de este experimento se han utilizado alrededor de 1300 biografías, similar al conjunto de datos inicial. A continuación en la figura 7.20 se muestra el dataset balanceado. Podemos observar como las clases inicialmente menos representadas aparecen ahora en mayor medida, haciendo a este dataset un conjunto con categorías balanceadas.

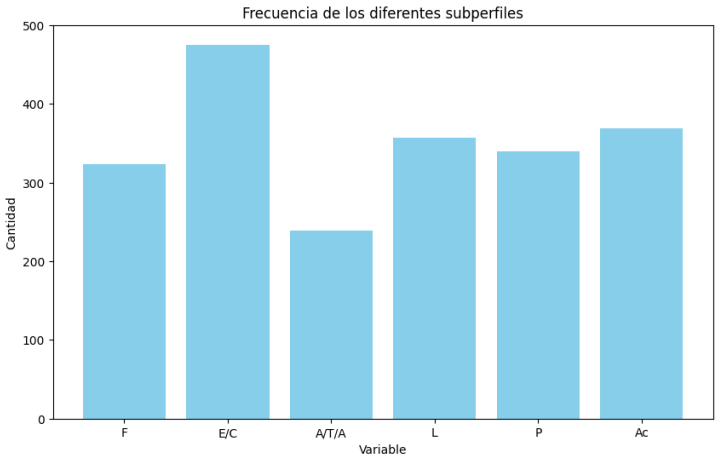


Figura 7.20: Conjunto de datos balanceado (Experimento 2)

Este experimento ha seguido la misma estructura que los anteriores. Se ha entrenado durante un total de 250 épocas con una función de pérdida MSE y un optimizador Adam. En la figura 7.21 se muestra una evolución de la función de pérdida para el conjunto de entrenamiento y validación.

A continuación se realiza la evaluación del modelo para las tres métricas utilizadas

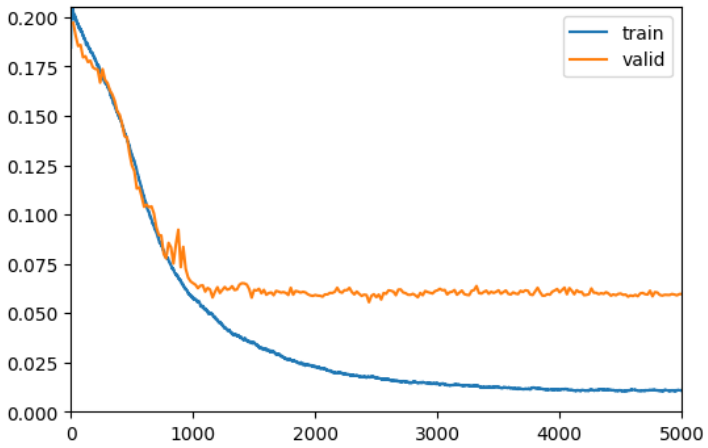


Figura 7.21: Evolución de la función de pérdida para el modelo AWD-LSTM Experimento 2 (Perfiles)

(MSE, MAE, R2). En la tabla 7.11 se presentan los resultados obtenidos.

Métricas	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Media**
MSE	0.1223	0.1993	0.0609	0.1524	0.0431	0.0736	0.1086
MAE	0.2009	0.2858	0.0943	0.1827	0.0964	0.1429	0.1672
R2	0.3160	0.1202	0.4128	0.3955	0.4668	0.2644	0.3293

Cuadro 7.11: Valores de MSE, MAE y R2 para diferentes perfiles (Experimento 2)

- En cuanto al MSE podemos observar un MSE medio de 0.108 algo superior que para el modelo base y el experimento 1. Todas la categorías presentan valores similares, a excepción de E/C y P, que presentan errores inusualmente altos y bajos, respectivamente. Como se ha ido viendo en los modelos anteriores.
- Observando la métrica MAE obtenemos un valor medio de 0.1671. Lo que indica que la predicción total se desvía de la realidad un 16.71 %. Un valor algo superior a la media tanto del modelo base como el experimento 1.
- En cuanto al R2 obtenemos un valor medio de 0.329, es decir, este modelo explica un 32.9 % de la variabilidad del modelo. Podemos encontrar valores bastante más homogéneos que en comparación con el modelo base y el experimento 1. A excepción del perfil E/C que presenta un valor bastante bajo en comparación al resto de perfiles.

A continuación, en la figura se muestra una comparativa de las diferentes métricas para el modelo base, el experimento 1 y 2.

En la siguiente tabla se muestra la primera parte de la validación del modelo. En esta parte se trata de ver como este modelo con los perfiles balanceados imita al experto humano.

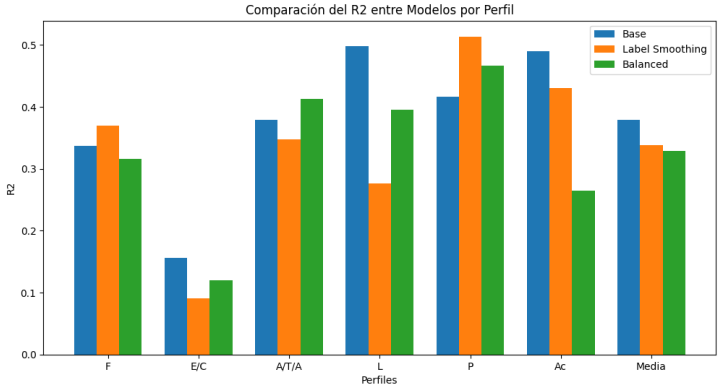


Figura 7.22: Comparativa R2 Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

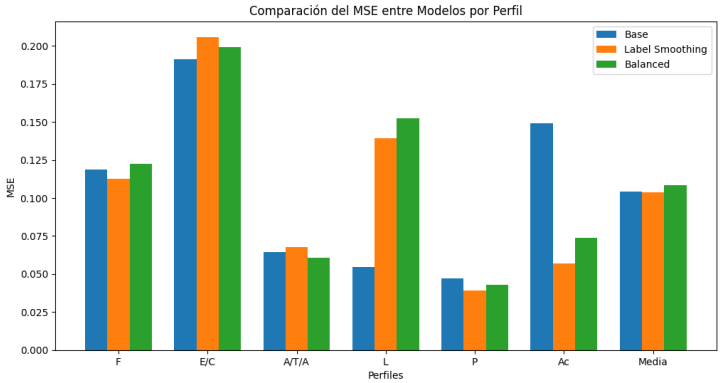


Figura 7.23: Comparativa MSE Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

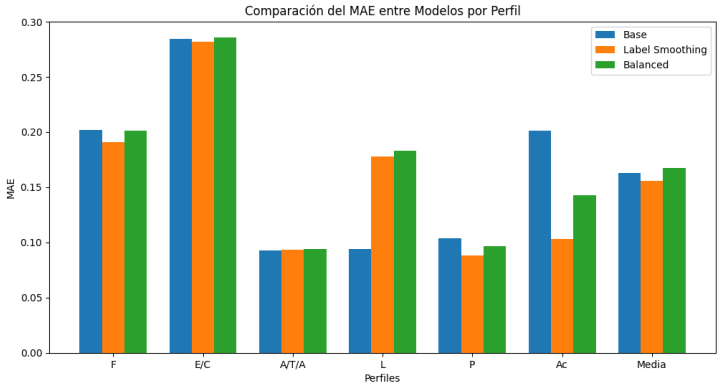


Figura 7.24: Comparativa MAE Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

- En cuanto al error encontramos un error bastante cercano al 0. Lo que indica una buena estimación entre las predicciones y los valores reales de los perfiles. Vemos que las categorías F, A/T/A y Ac tienen medias negativas, lo que indica que el modelo tiende a subestimar estos perfiles. Lo contrario pasa para las categorías E/C, L y P, donde el modelo las sobrestima.

- Observando la columna del error σ vemos que los valores son bastante homogéneos. A excepción de E/C donde tiene error σ bastante alto. Lo que indica que la dispersión para esta categoría es relativamente alta, indicando así que el modelo es más inconsistente en este perfil en comparación con el resto.

Perfil	μ (error)	σ (error)	corr. coef. (R)	std. error of μ
F	-0.074307	0.341686	0.627918	0.033667
E/C	0.071749	0.440633	0.521190	0.043417
A/T/A	-0.047576	0.242203	0.662413	0.023865
L	0.039205	0.388348	0.345719	0.038265
P	0.000195	0.207525	0.695948	0.020448
Ac	-0.004864	0.271262	0.544399	0.026728

Cuadro 7.12: Resumen de errores por perfil profesional experimento 2

- Observado los coeficientes de correlación vemos que están bastante ligados al valor R2 de la tabla anteriormente mencionada, ya que este coeficiente mide la correlación entre las observaciones reales y predichas para cada perfil. Se observan valores medios a excepción del perfil L.
- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

A continuación se muestra la segunda parte de la validación. Se realiza la discretización de las de las categorías principales en las cuatro categorías definidas anteriormente. En la tabla 7.13 se muestran los resultados.

Perfil	F	E/C	A/T/A	L	P	Ac
	67.961165	62.135922	86.407767	78.640777	81.553398	75.728155

Cuadro 7.13: Precisión del modelo AWD-LSTM para perfiles (Experimento)

Observando la tabla de precisión del modelo para el experimento 2 observamos una precisión media del 75.4 %. En particular destacan la precisión de los perfiles F, E/C, donde las precisiones son relativamente bajas. A continuación, en la figura 7.25 se puede observar una comparativa con el modelo base y el experimento 1 de este modelo. Observamos que no se han producido mejoras significativas en ninguno de los perfiles.

Experimento 2: AWD-LSTM subperfiles

Para la realización de este experimento se han utilizado alrededor de 1300 biografías, similar al conjunto de datos inicial. A continuación en la figura 7.26 se muestra el dataset balanceado. Podemos observar como las clases inicialmente menos representadas aparecen ahora mayor medida haciendo a este dataset un conjunto con subperfiles balanceados.

Este experimento ha seguido la misma estructura que los anteriores. Se ha entrenado durante un total de 250 épocas con una función de pérdida MSE y un optimizador Adam. En la figura 7.27 se muestra la evolución de la función de pérdida para el conjunto de entrenamiento y validación.

A continuación se realiza la evaluación del modelo AWD-LSTM de subperfiles en el experimento 2 para las tres métricas utilizadas (MSE, MAE, R2). En la tabla 7.14 se

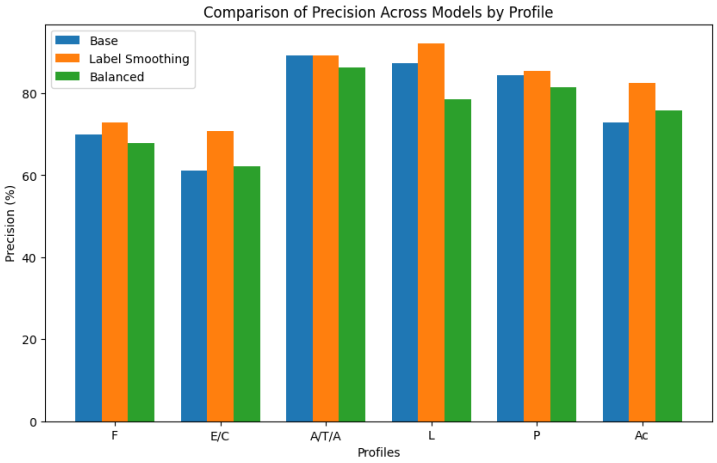


Figura 7.25: Comparativa de precisión del modelo AWD-LSTM (perfiles) modelo base, experimento 1 y experimento 2

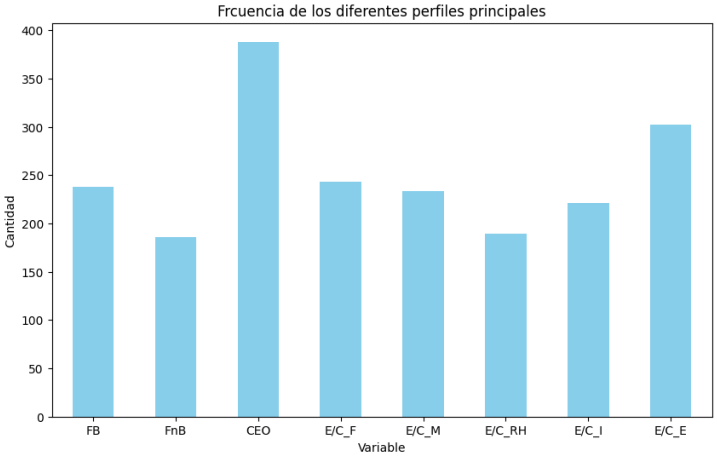


Figura 7.26: Conjunto de datos balanceado (Subperfiles)

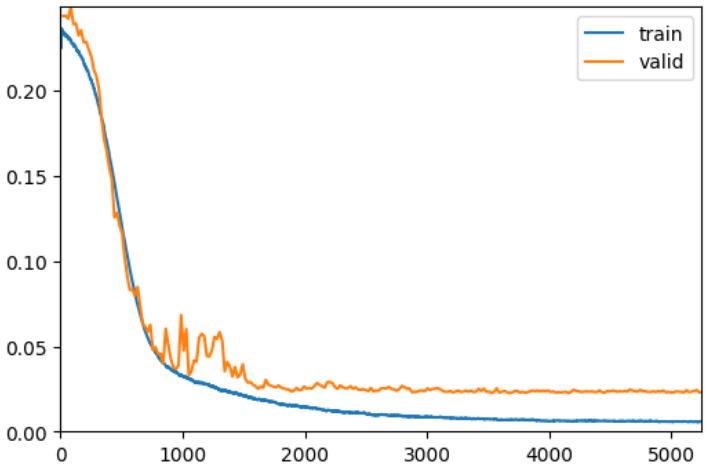


Figura 7.27: Evolución de la función de pérdida para el modelo AWD-LSTM Experimento 2 (Subperfiles)

presentan los resultados obtenidos.

Métricas	FB	FnB	CEO	E/C_F	E/C_M	E/C_RH	E/C_I	E/C_E	Media
MSE	0.0117	0.0110	0.0743	0.0287	0.0297	0.0249	0.0224	0.0638	0.0333
MAE	0.0479	0.0505	0.1473	0.0505	0.0577	0.0480	0.0534	0.1082	0.0704
R2	0.8788	0.8118	0.1257	0.6680	0.4670	0.4485	0.5961	0.2516	0.5309

Cuadro 7.14: Valores de MSE, MAE y R2 para diferentes subperfiles (Experimento 2)

- En cuanto al MSE podemos observar un MSE medio de 0.03 algo superior que para el modelo en el experimento 1, pero menor que el modelo base. Todas la categorías presentan valores relativamente bajos , a excepción de CEO y E/C_E que presentan errores inusualmente altos.
- En cuanto al MAE obtenemos un valor medio de 0.07. Lo que indica que la predicción total se desvía de la realidad un 7 %. Un valor algo inferior a la media del modelo base, pero algo superior que la media del experimento 1.
- En cuanto al R2 obtenemos un valor medio de 0.5309, es decir que este modelo explica un 53.09 % de la variabilidad del modelo. Podemos encontrar valores bastante más homogéneos que en comparación con el modelo base y el experimento 1. A excepción del perfil CEO que presenta un valor bastante bajo en comparación al resto de perfiles. Además, se han visto una mejora en la variabilidad del de los perfiles menos representados en el conjunto del modelo base y experimento 1 (E/C_M y E/C_H).

En las siguientes figuras se puede encontrar una comparativa donde las tres métricas evaluadas donde se puede visualizar mejor los resultados explicados anteriormente.

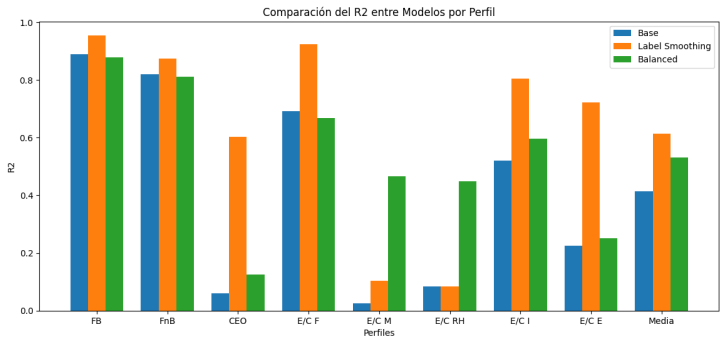


Figura 7.28: Comparativa R2 Modelo base, Experimento 1 y 2 (AWD-LSTM) subperfiles

En la siguiente tabla 7.15 se muestra la primera parte de la validación del modelo. En esta parte se trata de ver como este modelo con los perfiles balanceados imita al experto humano.

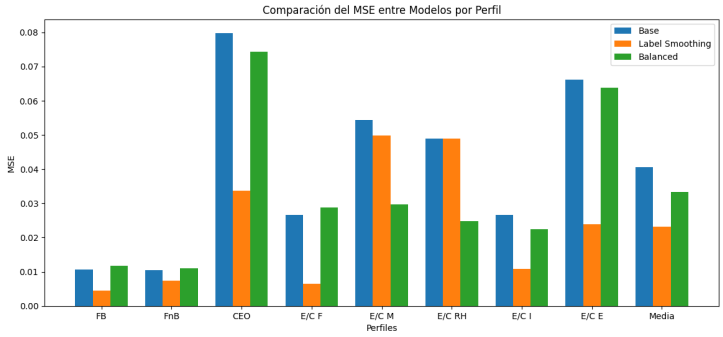


Figura 7.29: Comparativa MSE Modelo base, Experimento 1 y 2 (AWD-LSTM) subperfiles

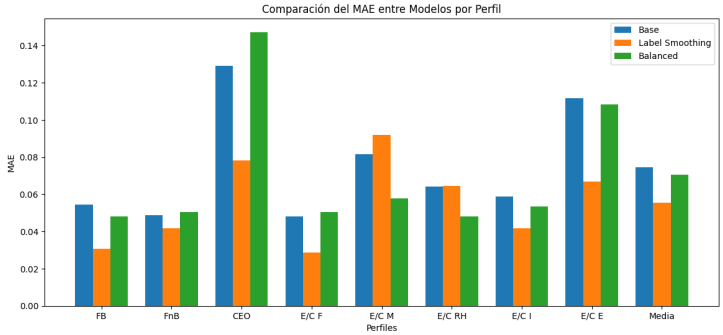


Figura 7.30: Comparativa MAE Modelo base, Experimento 1 y 2 (AWD-LSTM) subperfiles

Subperfil	μ **(error)**	σ **(error)**	**corr. coef. (R)**	**std. error of** μ
FB	0.004789	0.108016	0.939058	0.011580
FnB	0.003304	0.104888	0.903176	0.011245
CEO	0.107379	0.250519	0.692274	0.026858
E/C_F	-0.013677	0.168978	0.825150	0.018116
E/C_M	-0.017074	0.171359	0.696244	0.018372
E/C_RH	-0.024880	0.155755	0.680016	0.016699
E/C_I	-0.003797	0.149707	0.779931	0.016050
E/C_E	-0.075156	0.241155	0.571176	0.025855

Cuadro 7.15: Resumen de errores por subperfil AWD-LSTM experimento 2

- En cuanto al error encontramos un error bastante cercano al 0 en todos los subperfiles, a excepción del los subperfiles CEO y -0.075156 , que presenta un error cercano a 0.1. Lo que indica, que en general se realizan una buena estimación entre las predicciones y los valores reales de los subperfiles. Vemos que los subperfiles E/C_F, E/C_M, E/C_RH, E/C_I y E/C_E tienen medias negativas, lo que indica que el modelo tiende a subestimar estos perfiles. Lo contrario pasa para las categorías FB, FnB y CEO, donde el modelo las sobreestima.

- Observando la columna del error σ vemos que los valores son bastante homogéneos. A excepción de CEO y E/C_E donde tiene error σ bastante alto. Lo que indica que la dispersión para esta categoría es relativamente alta, indicando así que el modelo

es más inconsistente en este perfil en comparación con el resto.

- Observado los coeficientes de correlación vemos valores bastante altos para todos los subperfiles.
- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

A continuación se muestra la tabla 7.16 con la precisión de los diferente subperfiles para los valores discretizados en 4 categorías. Se observa una precisión bastante alta para todos los subperfiles con una precisión media del 88.42 %. En la figura 7.31 muestra una comparativa del modelo para el modelo base el experimento 1 y 2.

Perfil	**Precisión**
FB	89.655
FnB	89.655
CEO	74.713
E/C_F	90.8046
E/C_M	90.8046
E/C_RH	93.103
E/C_I	90.8046
E/C_E	78.161

Cuadro 7.16: Precisión del modelo AWD-LSTM Experimento 2 (Subperfiles)

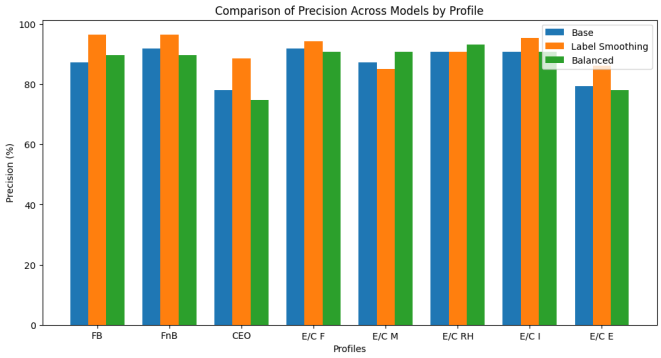


Figura 7.31: Comparativa de precisión del modelo AWD-LSTM (subperfiles) modelo base, experimento 1 y experimento 2

Se observa una precisión bastante igualada en todos los subperfiles para el modelo base, experimento 1 y experimento 2. Aunque se ha producido una mejora de la precisión de los superfiles menos representados en el conjunto de datos original (E/C_M, E/C_RH).

Experimento 2: Bert perfiles

Para la realización de este experimento se ha utilizado la misma metodología que para el resto de modelos del experimento. Se ha balanceado el conjunto de datos original obteniendo un total de 1300 biografías aproximadamente.

Se ha entrenado para un total de 50 épocas igual que el resto de modelos Bert. Durante las 10 primeras se ha realizado el congelamiento de la capas con un learning rate de 1.10-2. Posteriormente para las 40 restantes, se han descongelado las capas utilizando un learning rate 1.10-4. Como en el resto de modelos se ha utilizado una función de pérdida MSE y un optimizador Adam. A continuación en la figura 7.32 se muestra la evolución de la función de pérdida para el conjunto de datos de validación y entrenamiento del modelo Bert para categorías principales para el experiemento 2.

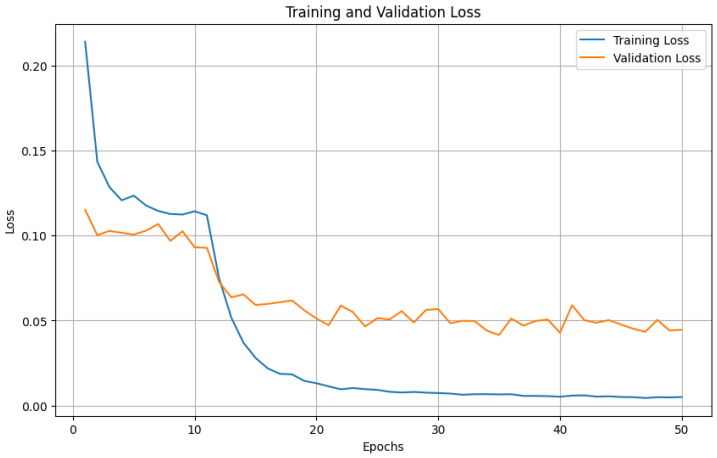


Figura 7.32: Evolución de la función de pérdida para el modelo Bert experimento 2 (Perfiles)

A continuación se realiza la evaluación del modelo para las tres métricas utilizadas (MSE, MAE, R2). En la tabla 7.17 se presentan los resultados obtenidos.

Métricas	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Media**
MSE	0.0753	0.1491	0.0444	0.0224	0.0429	0.0575	0.0653
MAE	0.1677	0.2352	0.0959	0.0419	0.0896	0.1082	0.1231
R2	0.5786	0.3419	0.5720	0.7952	0.4686	0.4257	0.5303

Cuadro 7.17: Valores de MSE, MAE y R2 para diferentes perfiles (Experimento 2)

- En cuanto al MSE podemos observar un MSE medio de 0.065 algo superior que para el modelo base y pero inferior al experimento 1. Todas la categorías presentan valores similares, a excepción de E/C , que presenta un error bastante alto.
- MAE obtenemos un valor medio de 0.123. Lo que indica que la predicción total se desvía de la realidad un 12.3 %. Un valor muy similar al obtenido en el modelo base y el experimento 1.

- En cuanto al R2 obtenemos un valor medio de 0.5303, es decir que este modelo explica un 53.03 % de la variabilidad del modelo. Podemos encontrar valores bastante más homogéneos que en comparación con el modelo base y el experimento 1. A excepción del perfil E/C que presenta un valor bastante bajo en comparación al resto de perfiles. Además encontramos una mejora en la variabilidad media explicada por el modelo con respecto al modelo base y el experimento 1.

A continuación se muestra la figura donde podemos visualizar las diferentes métricas para el modelo base, el experimento 1 y 2 del modelo Bert para los perfiles principales.

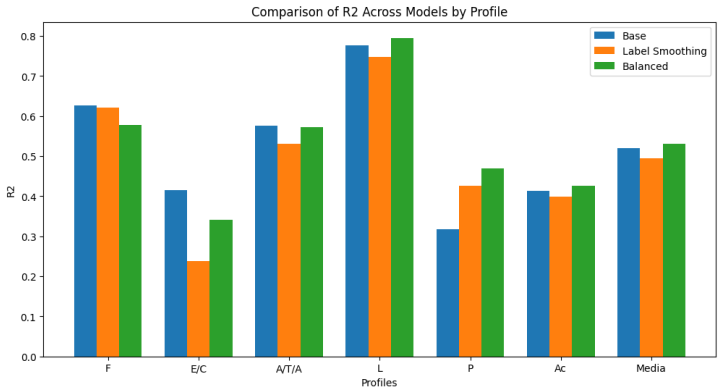


Figura 7.33: Comparativa R2 Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

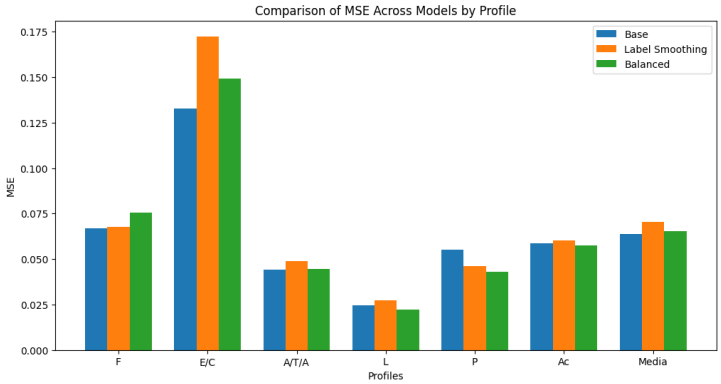


Figura 7.34: Comparativa MSE Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

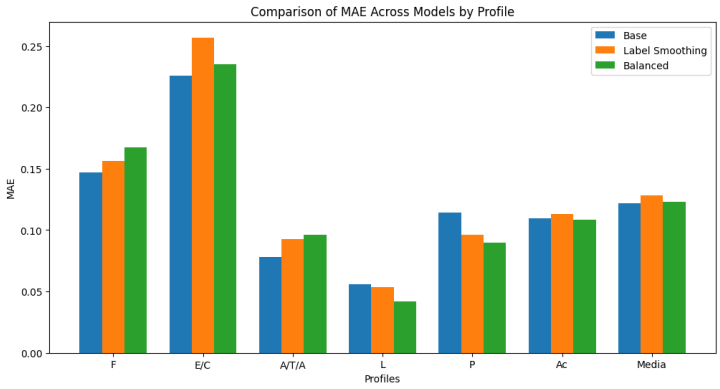


Figura 7.35: Comparativa MAE Modelo base, Experimento 1 y 2 (AWD-LSTM) perfiles

En la siguiente tabla 7.18 se muestra la primera parte de la validación del modelo. En esta parte se trata de ver como este modelo con los perfiles balanceados imita al experto humano.

Perfil	μ **(error)**	σ **(error)**	**corr. coef. (R)**	**std. error of** μ
F	0.032755	0.272494	0.773159	0.026850
E/C	0.135326	0.361617	0.674916	0.035631
A/T/A	0.012997	0.210338	0.765979	0.020725
L	-0.007854	0.149323	0.892253	0.014713
P	-0.030172	0.204970	0.699675	0.020196
Ac	0.028573	0.238008	0.708882	0.023452

Cuadro 7.18: Validación modelo Bert perfil experimento 2

- En cuanto al error encontramos un error bastante cercano al 0, a excepción del perfil E/C, que presenta un error superior a 0.1. Lo que indica una buena estimación entre las predicciones y los valores reales de los perfiles, a excepción del perfil E/C. Vemos que los perfiles L y P tienen medias negativas, lo que indica que el modelo tiende a subestimar estos perfiles. Lo contrario pasa para los perfiles restantes donde el modelo las sobrestima.

- Observando el error σ vemos que los valores son bastante homogéneos. A excepción de E/C y L donde tiene error σ bastante alto y bajo respectivamente. Lo que indica que en general la dispersión es normal, indicando así la consistencia del modelo en carácter general.

- Observado los coeficientes de correlación vemos que las correlaciones entre las observaciones reales y predichas para cada perfil son bastante elevadas, superiores al 65 %. Lo que indica una relación bastante fuerte entre las observaciones predichas y reales.

- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

En la tabla 7.19 se observa la precisión del modelo Bert para los perfiles principales para el experimento 2 con los valores discretizados en 4 categorías de acuerdo a los intervalos descritos anteriormente. Se observa una precisión general bastante alta del modelo con una precisión media del 84.02 %. Destacan los perfiles L y A/T/A con una precisión superior al 90 %.

Perfil	F	E/C	A/T/A	L	P	Ac
Precisión	70.873786	70.873786	90.291262	95.145631	86.407767	87.378641

Cuadro 7.19: Precisión del modelo Transformers para perfiles (Experimento)

A continuación, en la figura 7.36 se puede observar una comparativa con el modelo base, el experimento 1 y el experimento 2 del modelo Bert para perfiles. Observamos que método de balancear el dataset ha mejorado la precisión en todos los perfiles principales excepto el perfil financiero.

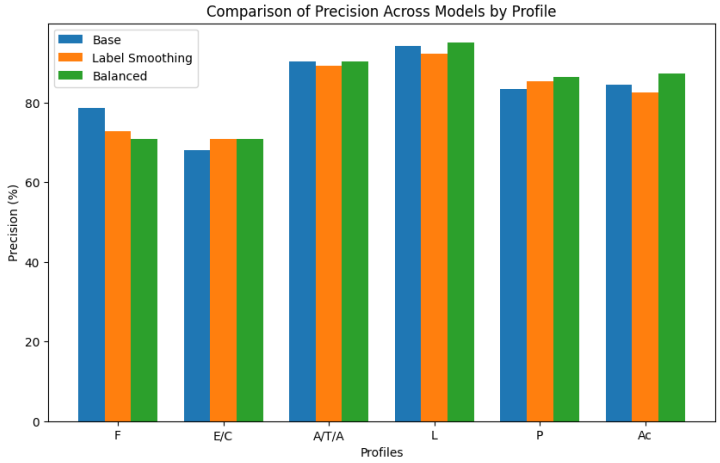


Figura 7.36: Comparativa de precisión del modelo Bert (perfiles) modelo base, experimento 1 y experimento 2

Experimento 2: Bert subperfiles

Para la realización de este experimento se han utilizado alrededor de 1300 biografías, similar al conjunto de datos inicial la distribución es la misma utilizada en el modelo LSTM para este mismo experimento, en la figura 7.26.

Este experimento ha seguido la misma estructura que los anteriores. Se ha entrenado un total de 50 épocas. Durante las primeras 10 épocas se ha entrenado con un *learning rate* 1.10-2, manteniendo las capas congeladas, para las 40 restantes se ha usado un *learning rate* de 1.10-4 descongelado las capas. Se ha usado una función de pérdida MSE y un optimizador Adam.

A continuación se realiza la evaluación del modelo Bert de subperfiles en el experimento 2 para las tres métricas utilizadas (MSE, MAE, R2). En la tabla 7.20 se presentan los resultados obtenidos.

Figura 7.37: Evolución de la función de pérdida para el modelo Bert Experimento 2 (Subperfiles)

Métricas	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**	**Media**
MSE	0.0071	0.0075	0.0299	0.0098	0.0105	0.0313	0.0260	0.0417	0.0205
MAE	0.0459	0.0347	0.0792	0.0442	0.0382	0.0512	0.0501	0.0966	0.0550
R2	0.9263	0.8719	0.6486	0.8862	0.8104	0.3053	0.5322	0.5113	0.6865

Cuadro 7.20: Valores de MSE, MAE y R2 para el modelo Bert subperfiles (Experimento 2)

- En cuanto al MSE podemos observar un MSE medio de 0.0204 bastante inferior a los valores obtenidos con el modelo base y el experimento 1. Todas la categorías presentan valores bajos, mejorando esta métrica en todos los subperfiles a excepción de CEO.
- En cuanto al MAE obtenemos un valor medio de 0.055. Lo que indica que la predicción total se desvía de la realidad un 5.5 %. Un valor algo inferior a la media del modelo base y experimento 1.
- En cuanto al R2 obtenemos un valor medio de 0.6865, es decir que este modelo explica un 68.65 % de la variabilidad del modelo. Encontramos una mejora en todos los subperfiles con respecto al modelo base a excepción del subperfil CEO. Además, se han visto una mejora sustancial en la variabilidad explicada del de los subperfiles menos representados en el conjunto del modelo base (E/C_F, E/C_M y E/C_RH, E/C_I y E/C_E).

En las siguientes figuras se puede encontrar una comparativa de las tres métricas evaluadas donde se puede visualizar mejor los resultados explicados anteriormente.

En la siguiente tabla se muestra la primera parte de la validación del modelo. En esta parte se trata de ver como este modelo con los perfiles balanceados imita al experto humano.

- En cuanto al error encontramos un error bastante cercano al 0 en todos los subperfiles. Lo que indica, que el modelo Bert con subperfiles para el experimento 2 realiza

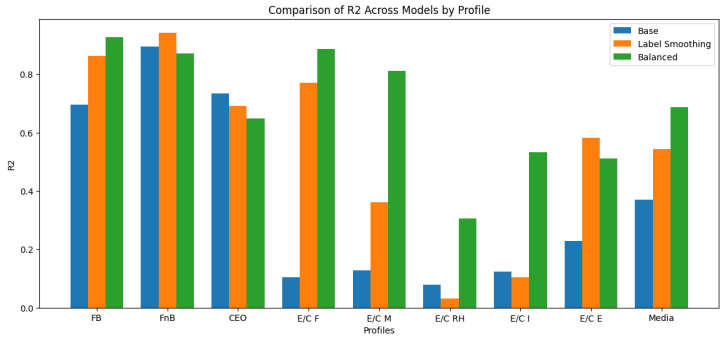


Figura 7.38: Comparativa R2 Modelo base, Experimento 1 y 2 (Bert) subperfiles

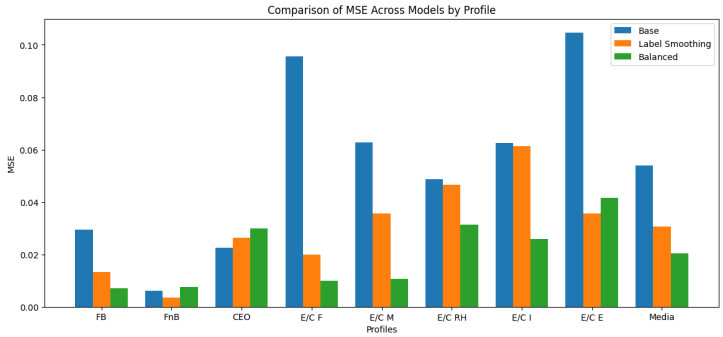


Figura 7.39: Comparativa MSE Modelo base, Experimento 1 y 2 (Bert) subperfiles

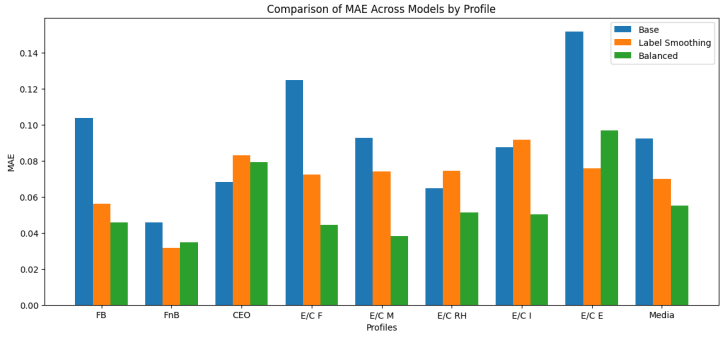


Figura 7.40: Comparativa MAE Modelo base, Experimento 1 y 2 (Bert) subperfiles

Subperfil	μ **(error)**	σ **(error)**	**corr. coef. (R)**	**std. error of** μ
FB	-0.002605	0.084280	0.963360	0.009036
FnB	-0.012591	0.085661	0.935825	0.009184
CEO	0.016733	0.171980	0.841953	0.018438
E/C_F	0.012994	0.098386	0.942528	0.010548
E/C_M	0.008678	0.102336	0.903037	0.010972
E/C_RH	-0.034339	0.173674	0.575725	0.018620
E/C_I	-0.018355	0.160114	0.737221	0.017166
E/C_E	0.003476	0.204095	0.732930	0.021881

Cuadro 7.21: Resumen de errores por subperfil profesional experimento 2 (Bert)

una buena estimación entre las predicciones y los valores reales de los subperfiles. Vemos que los subperfiles FB, FnB, E/C_RH, E/C_I tienen medias negativas, lo que indica que el modelo tiende a subestimar estos perfiles. Lo contrario pasa para el resto de subperfiles, donde el modelo las sobrestima.

- Observando la columna del error σ vemos que los valores son bastante homogéneos. A excepción de FB y FnB que presentan un error inusualmente bajo. Lo que indica que en general la dispersión del modelo es relativamente baja para el total de los subperifles y en especial para los subperfiles FB y FnB.
- Observado los coeficientes de correlación vemos valores extremadamente altos en todos los subperfiles a excepción de E/C_RH.
- Vemos que los valores son muy próximos a cero en todos los perfiles. Esto indica un alto grado de precisión en la estimación de la media en todos los perfiles.

A continuación se muestra la tabla 7.22 con la precisión de los diferente subperfiles para los valores discretizados en 4 categorías. Se observa una precisión bastante alta para todos los subperfiles con una precisión media del 89.24 %, algo superior a la precisión obtenida con el mismo modelo para los subperfiles en el modelo base. En la figura 7.41 muestra una comparativa del modelo para el modelo base y el experimento 1 y 2.

Perfil	**Precisión**
FB	88.5057
FnB	91.9540
CEO	81.6092
E/C_F	94.2529
E/C_M	93.1034
E/C_RH	90.8046
E/C_I	89.6552
E/C_E	85.0575

Cuadro 7.22: Precisión del modelo AWD-LSTM Experimento 2 (subperfiles)

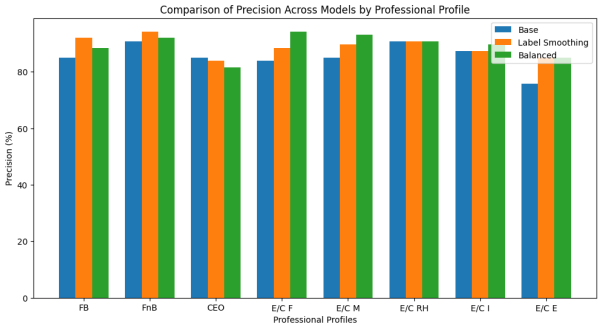


Figura 7.41: Comparativa de precisión del modelo Bert (subperfiles) modelo base, experimento 1 y experimento 2

Se observa una precisión bastante igualada en todos los subperfiles para el modelo en los experimentos 1 y 2. Cabe destacar que con el balanceo de los subperfiles del conjunto de datos se ha obtenido una mejora sustancial en 6 de los 8 subperfiles, a excepción del subperfil CEO, donde la precisión a bajado a un 81 %, y el subperfil E/C_RH donde la precisión se ha mantenido constante.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Tras el análisis de los resultados obtenidos para los modelos *AWD-LSTM* y *Bert* con la aplicación de técnicas como *label smoothing* y balanceo del conjunto de datos se presentan las siguientes conclusiones. Cabe destacar que las conclusiones extraídas se han basado en la precisión del modelo para las cuatro clases creadas, debido a que nuestro objetivo es la categorización de perfiles y subperfiles y no ver como el modelo imita el etiquetado humano, ya que como hemos mencionado éste está sujeto a la subjetividad humana.

En primer lugar, para los modelos base se ha encontrado que el modelo *Bert* tiene una mayor precisión en la categorización de las biografías de los consejeros independientes en los seis perfiles principales. En cuanto a la categorización en los diferentes subperfiles el modelo *AWD-LSTM* presenta una mayor precisión.

Podemos destacar que la técnica de *label smoothing* ha producido mejoras en 3 de los 4 modelos. El único modelo donde no se han producido mejoras en la precisión es el modelo *Bert* para las categorías principales, lo que indica que la técnica de *label smoothing* no tuvo el mismo efecto positivo en este caso particular. Los resultados obtenidos con la utilización de esta técnica sugieren que es una estrategia efectiva para mejorar la precisión de nuestros modelos, aunque su utilización debe evaluarse en cada caso, ya que puede tener un impacto diferente dependiendo del modelo y la tarea específica.

Por otro lado, al balancear el conjunto de datos, se han encontrado resultados similares. En este caso, la técnica también generó mejoras en 3 de los 4 modelos, siendo el modelo *AWD-LSTM* para los seis perfiles principales el que no presentó una mejora en la precisión.

Estos hallazgos sugieren que tanto la técnica de *label smoothing* como el balanceo del dataset son estrategias efectivas para mejorar la precisión de nuestros modelos. Sin embargo, es importante destacar que su impacto puede variar dependiendo del modelo y la tarea específica, tal como se observó en los casos excepcionales de los modelos *Bert* y *AWD-LSTM*.

8.2. Trabajo futuro

A raíz de estos resultados obtenidos en este trabajo de fin de grado, se presentan varias ideas para la implementación en un futuro trabajo. Hemos visto como las técnicas de *label smoothing* y el balanceo del conjunto de datos han demostrado ser efectivas a la hora de aumentar la precisión de los modelos.

En primer lugar se propone la idea de investigar otras técnicas de aumento de datos como la reescritura de las biografías, además de su implementación con las técnicas utilizadas en este trabajo y medir su impacto conjunto. Por otro lado, se presenta la idea de la utilización de otros modelos base más complejos que puedan llegar a dar mejores resultados, como el modelo de *RoBerta*.

Referencias

[1] «*fine_tune*» *vs.* «*fit_one_cycle*». fast.ai Course Forums, Accessed: 2024-06-25. Mar. de 2020. URL: https://forums.fast.ai/t/fine-tune-vs-fit-one-cycle/66029/3.

[2] Toolify AI. *¿Qué es el problema del gradiente explosivo en Deep Learning?* https://www.toolify.ai/es/ai-news-es/qu-es-elproblema-del-gradiente-explosivo-en-deep-learning-1122191. Recuperado el 21 de febrero de 2024. 2024.

[3] *Ajuste de un modelo con la API Trainer - Hugging Face NLP Course*. https://huggingface.co/learn/nlp-course/es/chapter3/3?fw=pt. n.d.

[4] Anna. *¿Cuánto tiempo dura un ordenador portátil? ¿Cuándo comprar un ordenador portátil nuevo?* abril de 2023. URL: https://www.minitool.com/es/respaldar-datos/cuanto-dura-un-ordenador-portatil.html.

[5] T. Asana. *Matriz de riesgos: cómo evaluar los riesgos para lograr el éxito del proyecto*. Asana. Feb. de 2024. URL: https://asana.com/es/resources/risk-matrix-template.

[6] AWD-LSTM. *AWD-LSTM*. s.f. URL: https://fastai.github.io/fastai-docs/text.models.awdlstm.

[7] Bankinter. *Criterios de las empresas del IBEX 35*. Consultado el: 28 de junio de 2024. Jun. de 2024. URL: https://www.bankinter.com/blog/mercados/criterios-empresas-ibex35.

[8] CaixaBank. *Consejo y consejeros*. 2024. URL: https://www.caixabank.com/deployedfiles/caixaholding/Estaticos/PDFs/CriteriaInstitute/ElAulaDelAccionista/aula770.pdf.

[9] N. V. Chawla et al. «SMOTE: Synthetic Minority Over-sampling Technique». En: *Journal of Artificial Intelligence Research* 16 (2002), págs. 321-357. DOI: 10.1613/jair.953. URL: https://doi.org/10.48550/arXiv.1106.1813.

[10] Keith Cochran., Clayton Cohn. y Peter Hastings. «Improving NLP Model Performance on Small Educational Data Sets Using Self-Augmentation». En: *Proceedings of the 15th International Conference on Computer Supported Education - Volume 1: CSEDU*. INSTICC. SciTePress, 2023, págs. 70-78. ISBN: 978-989-758-641-5. DOI: 10.5220/0011857200003470.

[11] Wikipedia contributors. *Regularization (mathematics)*. Accessed: 2024-06-25. Jun. de 2024. URL: https://en.wikipedia.org/wiki/Regularization_(mathematics).

[12] DeepLearning.AI. *Natural Language Processing (NLP) [A complete guide]*. https://www.deeplearning.ai/resources/natural-language-processing/. Accessed: 2024-06-25. Ene. de 2023.

[13] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *CoRR* (2018). arXiv: 1810.04805.

[14] ESADE. *La relevancia de la independencia en los Consejos de Administración del IBEX-35*. Informe de investigación. 2024. URL: https://www.esade.edu/itemsweb/wi/research/cgc/Informe_La_relevancia_de_la_independencia_en_los_CA_del_IBEX-35.pdf.

[15] *fastai - Data block*. Fastai, https://docs.fast.ai/data.block.html#regressionblock. n.d.

[16] *fastai - Text transfer learning*. Fastai, https://docs.fast.ai/tutorial.text.html. n.d.

[17] P. González. *Deep Learning*. Online. Disponible en: https://pglez82.github.io/DeepLearningWeb/s 2022.

[18] Google-Research. *GitHub - google-research/bert: TensorFlow code and pre-trained models for BERT*. Accessed: 2024-06-25. URL: https://github.com/google-research/bert?tab=readme-ov-file.

[19] Jeremy Howard y Sebastian Ruder. «Universal Language Model Fine-tuning for Text Classification». En: (2022). Jeremy Howard affiliated with fast.ai and University of San Francisco, Sebastian Ruder affiliated with Insight Centre, NUI Galway and Aylien Ltd., Dublin.

[20] Hugging face NLP course. *Tokenizadores - Hugging face NLP course*. https://huggingface.co/learn/nlp-course/es/chapter2/4. s.f.

[21] Interactive Chaos. *Sobreentrenamiento*. https://interactivechaos.com/es/wiki/sobreentrenamiento. n.d.

[22] Justin M. Johnson y Taghi M. Khoshgoftaar. «Survey on deep learning with class imbalance». En: *Journal of Big Data* 6.1 (2019), pág. 27. DOI: 10.1186/s40537-019-0192-5. URL: https://doi.org/10.1186/s40537-019-0192-5.

[23] J. Jordan. *Gradient descent*. [En línea; consultado el 22 de junio de 2024]. Jul. de 2018. URL: https://www.jeremyjordan.me/gradient-descent/.

[24] Sowmya V. Kulkarni, Anuj Shivananda y Harshit Bharadwaj. *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*. O'Reilly Media, 2020.

[25] A. Kumar. *MSE vs RMSE vs MAE vs MAPE vs R-Squared: When to Use?* abril de 2024. URL: https://vitalflux.com/mse-vs-rmse-vs-mae-vs-mape-vs-r-squared-when-to-use/.

[26] Dominic Masters y Carlo Luschi. «Revisiting Small Batch Training for Deep Neural Networks». En: *ArXiv* abs/1804.07612 (2018). URL: https://api.semanticscholar.org/CorpusID:5032969.

[27] Na Na. *Breve historia de las redes neuronales artificiales — Aprende Machine Learning*. Feb. de 2019. URL: https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/.

[28] OpenAI. *GPT-2 Model Card*. 2024. URL: https://huggingface.co/openai-community/gpt2?text=Once+upon+a+time%2C.

[29] PyTorch. *Data Loading and Processing Tutorial*. https://pytorch.org/tutorials/beginner/data_loading_tutorial.html. 2024.

[30] *Salarios de Ingeniero de Machine Learning*. https://es.indeed.com/career/machine-learning-engineer/salaries. Accedido el 29 de junio de 2024. Indeed, 2024.

[31] Robin M. Schmidt. «Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview». En: *arXiv preprint arXiv:1912.05911* (2019). URL: https://arxiv.org/pdf/1912.05911.

[32] D. Scientist. *Deep Learning - How Long Short-Term Memory Networks (LSTMs) work: A Visual Guide*. Deep Learning Nerds — The Ultimate Learning Platform For AI And Data Science. Feb. de 2024. URL: https://www.deeplearningnerds.com/deep-learning-how-lstms-work/.

[33] Lichao Sun et al. «Mixup-Transformer: Dynamic Data Augmentation for NLP Tasks». En: *Proceedings of the 28th International Conference on Computational Linguistics*. Ed. por Donia Scott, Nuria Bel y Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, dic. de 2020, págs. 3436-3440. DOI: 10.18653/v1/2020.coling-main.305. URL: https://aclanthology.org/2020.coling-main.305.

[34] Zhongbin Sun et al. «A novel ensemble method for classifying imbalanced data». En: *Journal of Computer Science & Technology* (2024). Dept. of Computer Science & Technology, Xi'an Jiaotong University, China 710049; Dept. of Computer Science & Technology, Nanjing University, China 210093.

[35] César Vaca y Benjamín Tejerina Fernando y Sahelices. «Board of Directors' Profile: A Case for Deep Learning as a Valid Methodology to Finance Research». En: *Journal of Financial Research* (2024).

[36] Ashish Vaswani et al. «Attention is All You Need». En: *arXiv preprint arXiv:1706.03762* 30 (2017), págs. 5998-6008. URL: https://arxiv.org/pdf/1706.03762v5.

[37] Colaboradores de Wikipedia. *Procesamiento de lenguajes naturales*. Wikipedia, la Enciclopedia Libre. abril 29 de 2024. URL: https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales.

[38] Wikipedia contributors. *PyTorch*. https://es.wikipedia.org/wiki/PyTorch. 2023.

[39] Wikipedia, la enciclopedia libre. *Error cuadrático medio*. Mayo de 2024. URL: https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio.

[40] Chang-Bin Zhang et al. «Delving Deep into Label Smoothing». En: *Journal of Machine Learning Research* 23.4 (2022), págs. 123-145. DOI: 10.1234/jmlr.2022.56789. URL: https://www.jmlr.org/papers/v23/zhang22a.html.

ANEXO

Vamos a definir las biografías utilizadas en las predicciones predicciones realizadas:

Biografías predicciones perfiles

bio_1: JUAN FRANCISCO GOMARIZ HERNÁNDEZ (Molina de Segura, 07/12/1973) Consejero Es consejero desde junio del 2015. Es miembro de la Comisión de Auditoría y de la Comisión de Nombramientos y Retribuciones. Licenciado en Derecho por la Universidad de Murcia y Máster en Asesoría Jurídica de Empresas por el Instituto de Empresa (Madrid). Su carrera profesional la ha desarrollado a lo largo de 17 años asesorando a empresas de diversos sectores y principalmente del Sector de las Telecomunicaciones, TIC‘s y Audiovisual. Actualmente es Socio Director de Nexo Abogados, y Socio de Pitotamo Inversiones, S.L. Anteriormente fue responsable de la Asesoría Jurídica interna de Corporación Industrial Playa, S.A. (Almería), para pasar más tarde a formar parte de la Asesoría Jurídica interna de Vodafone España, S.A.U. (MadridValencia), desempeñando diversas funciones dentro de la misma entre las que destacó la responsabilidad de la asesoría jurídica en la Gerencia de LevanteBaleares.

bio_2: Javier Iglesias de Ussel y Ordís cuenta con una dilatada carrera en el mundo financiero. En 1974 se incorporó en Londres a Lloyds Bank International donde a lo largo de 21 años tuvo diversos puestos de responsabilidad en Banca Corporativa en Dubai, Sao Paulo, Asunción y Madrid. En 1995 se incorporó a The Bank of New York y fue nombrado Country Manager para la Península Ibérica. En 2002 se trasladó a Nueva York y fue nombrado Director General para América Latina. Del 2008 hasta diciembre del 2013 dirigió la Oficina de Representación del banco chileno Banco de Crédito e Inversiones. Desde 2008 el Sr. Iglesias de Ussel es Consejero Independiente en Inmobiliaria Colonial y desde marzo de 2015 es también Consejero Independiente en Aresbank. El Sr. Iglesias de Ussel es Licenciado en Historia Moderna por la Universidad de Barcelona y a lo largo de su carrera profesional ha participado en un gran número de cursos en Dirección y Administración de Empresas, Marketing, Análisis de Riesgo y Prevención contra el Lavado de Dinero. Ha vivido 22 años fuera de España y habla inglés, francés y portugués.

bio_3: Es licenciado en Derecho por la Universidad Complutense de Madrid. Realizó estudios de Economía en dicha Universidad y Fiscalidad en el Centro de Estudios Económicos y Tributarios. Ha sido empresario, consultor y consejero en diversas empresas. Fue secretario general del Sindicato Empresarial Alavés (SEA) de 1979 a 1995. Fue secretario general de la Confederación de Empresarios Vascos (Confebask) desde octubre de 1995 a marzo de 2011. Ha sido miembro de las Juntas Directivas u Órganos de Gobierno de

las principales Instituciones Socio Económicas del País Vasco entre otras de la Sociedad para la Promoción y Reconversión Industrial, de los Consejos Económico y Social y de Relaciones Laborales. Ha sido miembro del Consejo Económico y Social de España y de sus Comisiones de Economía y de Relaciones Laborales. Asimismo, ha sido miembro de la Comisión Permanente de la Escuela Andaluza de Economía y ha participado en diversos cursos y conferencias en la Universidad Internacional Menéndez Pelayo, los cursos de verano de El Escorial y la universidad de verano de la Universidad de País Vasco.

bio_4:El Sr. Peña Pinto es licenciado en Derecho por la Universidad Complutense de Madrid, habiendo superado la oposición a Inspector Técnico de Trabajo y de la Seguridad Social. Del año 1984 al 1989, el Sr. Peña ocupó el cargo de Consejero Laboral en la Embajada de España en Italia; posteriormente, entre 1991 y 1996, desempeñó los puestos de Secretario General de Salud dentro del Ministerio de Sanidad y Consumo y de Secretario General de Empleo y Relaciones Laborales dentro del Ministerio de Trabajo. Entre los años 2005 y 2006, fue designado Consejero Experto del Consejo Económico y Social, organismo que ha presidido hasta el mes de abril de este año. Asimismo, el Sr. Peña Pinto ha sido Consejero Nato del Consejo de Estado, cargo vinculado al puesto de presidente del Consejo Económico y Social. En abril de 2020, el Sr. Peña fue nombrado patrono de la Fundación CEOE. Por lo que se refiere a otras actividades profesionales desempeñadas, cabe destacar que don Marcos Peña está especializado en negociación colectiva de trabajo, habiendo ejercido el cargo de Presidente de la Comisión Negociadora de múltiples convenios colectivos (p. ej., Telefónica, RENFE, Repsol, Alcatel, Endesa, Astilleros, etc.). Además, el Sr. Peña Pinto es árbitro y mediador de distintos conflictos laborales de dimensión nacional, así como autor de numerosas publicaciones y articulista habitual en prensa escrita. Fue nombrado consejero independiente de la Sociedad por cooptación y aceptó su nombramiento el 9 de mayo de 2019, siendo reelegido por la Junta General ordinaria de 12 de junio de 2019.

Biografías predicciones subperfiles

bio_1: Licenciado en Administración de Empresas y MBA por ESADE, Barcelona. Actualmente, es Consejero Coordinador de SPS, socio de Aliqua Consulting y el consultor de empresas especializado en consultoría financiera y estratégica. Es fundador de Credit Risk Classification Group, dedicada a proporcionar asesoramiento y outsourcing en el ámbito del riesgo de crédito para empresas. Inició su carrera profesional como analista en el área de Business intelligence (Software AG) y como Subdirector responsable de operaciones y estrategia en Assistencial Club Mutual de Conductors, y también como analista en desarrollo de negocio y en diversas posicionesde management en American International Group (AIG), tanto en España, Europa y en Estados Unidos.

bio_2: En la actualidad es Consejero Delegado y Socio de VENTURE FINANZAS, S.V., S.A., Consejero de ZUNZUNEGUI HERMANOS, S.A., PATENTES Y MARCAS, INVERSIONES RIOCOBO, S.A., SICAV, E.S.F. EURORENT, SICAV, S.A., y B.S. INVERSIONES 97, SICAV, S.A., Secretario de LAR MAC 02 SICAV, S.A., y Presidente de VILLCAD BOLSA, SICAV, S.A.

bio_3: ECONOMISTA Licenciado en Económicas por la Universidad de Buenos Aires, en Relaciones Internacionales por la Universidad del Salvador, Curso de Administración de Empresas para Directivos por la Universidad Católica Argentina, y especializado en Mercado de futuros en Escuela de Economía de Londres y en Mercado de Capitales DeanWitter, en Nueva York. A su destacada trayectoria como Consejero de Grupo SANJOSE se une su experiencia en los Consejos de diversas compañías en las que desarrolla su labor actualmente Consejero de la Agencia de Bolsa Aldazabal y Cia desde 1980, Consejero de Mapfre Argentina Seguros Generales desde 1998, Consejero de Mapfre riesgos del trabajo (Mutua) desde 2000, Consejero de Carlos Casado desde 2008, Consejero de Pamsa desde 2009 hasta la fecha, Consejero de la Bolsa de Comercio de Buenos Aires desde 2004 y miembro de la mesa directiva y Consejero del Grupo Boldt desde 2005. Con anterioridad también fue Consejero de Metrogas (controlada por Repsol desde 2002 hasta el 2008) y Consejero de Banco Caudal (representante de Banco de Inversión DeanWitter).

bio_4:Licenciado en Derecho y Ciencias Empresariales (E3) por ICADE, es socio fundador de la empresa Tasmania Gestión. En el año 2000 fue también fundador de la sociedad financiera N+1 y ha sido consejero de Ezentis, Funespaña, General de Alquiler de Maquinaria (GAM) y Campofrío, entre otras sociedades. En su trayectoria profesional ha sido también director de Mercados de Capitales de AB Asesores Bursátiles, socio de Morgan Stanley y auditor de Arthur Andersen.

ANEXO A: Modelo LSTM predicciones 10 biografías (Perfiles)

Biografía	**F**	**E/C**	**A/T/A**	**L**	**P**	**Ac**	**Tipo**
bio_1	0.0	1.0	0.0	1.0	0.0	0.0	Etiquetado
	0.048404377	0.1753136	0.07648868	0.9602397	0.045973744	0.10049713	Base
	0.2276	0.67578	0.006798	0.041626	0.051941	0.03582	Exp.1
	0.75537	0.5229	0.04553	0.016922	0.02333	0.02014	Exp.2
bio_2	0.8	0.0	0.0	0.0	0.0	0.0	Etiquetado
	0.4560	0.3150	0.003	0.0261	0.0089	0.01596	Base
	0.5068	0.5625	0.001192	0.001086	0.05728	0.03790	Exp.1
	0.3979	0.6489	0.000257	0.00300	0.023239	0.02049	Exp.2
bio_3	0.0	0.8	0.0	0.0	0.4	0.0	Real
	0.00988	0.031639	8.2438e-05	0.018152	0.015742	0.00392	Base
	0.0005	0.00257	0.00322	0.01155	0.0874	0.00129	Exp.1
	0.00103	0.0005	0.0007	0.01348	0.02954	0.0020	Exp.2
bio_4	0.0	0.0	0.0	1.0	1.0	0.0	Real
	0.0355	0.72790	0.0019	0.55961	0.93343	0.005439	Base
	0.00215	0.925	0.0004	0.0596	0.83447	0.00071	Exp.1
	0.01634	0.2359	0.0025	0.084	0.9638	0.00140	Exp.2

ANEXO B: Modelo LSTM predicciones 10 biografías (Subperfiles)

Biografía	**FB**	**FnB**	**CEO**	**E/C_F**	**E/C_M**	**E/C_RH**	**E/C_I**	**E/C_E**	**Tipo**
bio_1	0.0	0.7	0.0	0.5	0.0	0.0	0.0	0.5	Etiquetado
	0.1365	0.85428	0.01343	0.8146	0.055	0.04458	0.13293	0.282	Base
	0.068	0.773	0.0035	0.731	0.0106	0.0066	0.0886	0.3701	Exp.1
	0.034	0.878	0.018	0.637	0.0687	0.0082	0.226	0.34008	Exp.2
bio_2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	Etiquetado
	0.0024	0.9833	0.00835	8.114e-05	1.79e-05	3.021e-05	9.374e-05	4.331e-05	Base
	0.0098	0.988	0.0008	0.0001	5.31e-05	3.82e-05	0.00014	8.416e-05	Exp.1
	0.007	0.996	0.0127	0.00196	2.84e-05	1.186e-05	6.503e-05	2.23e-05	Exp.2
bio_3	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	Real
	0.00485	0.820	0.0001	6.238e-05	0.00026	3.97e-05	0.00376	0.0003	Base
	0.0096	0.847	0.0012	6.354e-05	0.00016	1.847e-05	0.00256	0.00036	Exp.1
	0.0158	0.7275	0.07	0.00017	0.00186	0.0004	0.0029	0.00166	Exp.2
bio_4	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0	Real
	0.06	0.8741	0.012	0.005	0.0005	0.0003	0.0007	0.00135	Base
	0.1367	0.895	0.0008	0.0034	0.0013	0.0001	0.0047	0.0018	Exp.1
	0.085	0.953	0.0098	0.022	0.0017	0.0004	0.0040	0.01467	Exp.2

ANEXO C: Modelo Bert predicciones 10 biografías (Perfiles)

Biografía	F	E/C	A/T/A	L	P	Ac	Tipo
bio_1	0.0	1.0	0.0	1.0	0.0	0.0	Etiquetado
	0.01827	0.41814	0.014017	0.88543	0.0901	0.14136	Base
	0.012647	0.4065	0.050724	0.871621	0.010904	0.081579	Exp.1
	0.04840	0.17531	0.07648	0.96023	0.0459737	0.10049	Exp.2
bio_2	0.8	0.0	0.0	0.0	0.0	0.0	Etiquetado
	0.986975	0.00292	0.017285	0.012666	0.018482	0.0057	Base
	0.99493	0.03327	0.008726	0.005138	0.009257	0.00390	Exp.1
	0.98982	0.013740	0.0157	0.00670	0.01099	0.016724	Exp.2
bio_3	0.0	0.8	0.0	0.0	0.4	0.0	Real
	0.04992	0.03615	0.00607	0.07285	0.10319	0.034224	Base
	0.4562	0.99475	0.065845	0.000703	0.1119	0.32447	Exp.1
	0.5064	0.9544	0.058729	0.003004	0.074034	0.29667	Exp.2
bio_4	0.0	0.0	0.0	1.0	1.0	0.0	Real
	0.0422	0.594	0.0158	0.0371	0.533	0.04616	Base
	0.0366	0.8638	0.00811	0.00557	0.8626	0.0270	Exp.1
	0.0553	0.580	0.00619	0.1437	0.9313	0.0239	Exp.2

ANEXO D: Modelo Bert predicciones 10 biografías (Subperfiles)

Biografía	FB	FnB	CEO	E/C_F	E/C_M	E/C_RH	E/C_I	E/C_E	Tipo
bio_1	0.0	0.7	0.0	0.5	0.0	0.0	0.0	0.5	Etiquetado
	0.4091	0.581	0.0633	0.0898	0.01853	0.0089	0.01308	0.02465	Base
	0.00916	0.8437	0.0571	0.7101	0.0189	0.02538	0.0410	0.2335	Exp.1
	0.0664	0.9097	0.1139	0.5453	0.07058	0.0244	0.0040	0.0724	Exp.2
bio_2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	Etiquetado
	0.02621	0.9194	0.00478	0.0314	0.0073	0.00348	0.0083	0.00573	Base
	0.0084	0.9455	0.00793	0.0198	0.001763	0.00402	0.01609	0.007602	Exp.1
	0.0124	0.9737	0.0254	0.01392	0.0240	0.00326	0.0002	0.00687	Exp.2
bio_3	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	Real
	0.04413	0.3898	0.0100	0.0198	0.0093	0.00496	0.00656	0.01278	Base
	0.02247	0.5325	0.02609	0.0142	0.003129	0.00580	0.0089	0.0049	Exp.1
	0.0279	0.3739	0.05096	0.0471	0.02901	0.01323	0.00607	0.0186	Exp.2
bio_4	0.2	0.8	0.0	0.0	0.0	0.0	0.0	0.0	Real
	0.48258	0.4609	0.0080	0.0856	0.0154	0.00842	0.0109	0.0257	Base
	0.0713	0.8343	0.03461	0.0322	0.00274	0.00560	0.0122	0.005869	Exp.1
	0.1002	0.752	0.008	0.05321	0.0062	0.0139	0.0019	0.0116	Exp.2