



Universidad Politécnica
de Madrid



Escuela Técnica Superior de
Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

**Resumen de textos literarios en
español por medio de modelos
lingüísticos *Transformers***

Guillermo Marco Remón
Tutor: Francisco Serradilla García

Madrid, 13 de julio de 2020

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial*

Título: Resumen de textos literarios en español por medio de modelos lingüísticos *Transformers*

13 de julio de 2020

Autor(a): Guillermo Marco Remón

Tutor(a): Francisco Serradilla García
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Los recientes modelos lingüísticos basados en atención, los llamados Transformers, en particular sus variantes preentrenadas como BERT (*Bidirectional Encoder Representations from Transformers*), han superado el rendimiento del estado del arte en varias tareas clásicas de Procesamiento de Lenguaje Natural, tales como los sistemas pregunta y respuesta, diálogo o clasificación. Este trabajo se propone estudiar su rendimiento en una de las tareas tradicionalmente más complejas: el resumen abstractivo. Con este fin, se ha elaborado un corpus etiquetado de textos literarios en español, aportación inédita cuya novedad y magnitud (se compone de más 40000 libros etiquetados por fecha, autor, resumen realizado por humanos...) abre un campo de posibilidades de aplicación en herramientas de aprendizaje profundo, imposibles hasta ahora en español. Se ha probado un modelo basado en el ajuste fino de BERT para resumen abstractivo; se ofrecen resultados aceptables pero insuficientes, debidos a las dificultades que presentan el entrenamiento de un modelo de cientos de millones de parámetros y, sobre todo, la dificultad de capturar las dependencias a largo plazo que se dan en secuencias de texto de gran longitud. No obstante, se ha probado su rendimiento en una tarea donde la entrada no supera el tamaño máximo del modelo preentrenado: la generación de sinopsis a partir de sus títulos; en ella se observa un interesante potencial de los Transformers para la generación de texto, pues la red generaliza la elaboración de una sinopsis de tal manera que crea argumentos coherentes para títulos de libros que no existen; se trata de una inteligencia artificial que presenta la capacidad de generalización de las reglas del lenguaje así como cierta creatividad; se ha habilitado un bot de Twitter para difundir sus resultados y que los usuarios puedan probar la generación de sinopsis. Por último, se establece un nuevo estado del arte de la métrica ROUGE de resumen sobre este corpus. Se proponen soluciones y trabajos futuros para mejorar los resultados.

Abstract

Recent attention-based linguistic models, the so-called Transformers, in particular their pre-trained variants like BERT (Bidirectional Encoder Representations from Transformers), have outperformed the state-of-the-art in several classical Natural Language Processing tasks, such as question and answer systems, dialogue or classification. This paper aims to study their performance in one of the traditionally most complex tasks: abstractive summarization. To this end, a labelled corpus of literary texts in Spanish has been elaborated. This is an unpublished contribution whose novelty and magnitude (it is composed of more than 40,000 books labelled by date, author, summary made by humans...) opens a field of application possibilities in deep learning tools, impossible until now in Spanish. A model based on BERT's fine-tuning for abstractive summaries has been tested; it offers acceptable but insufficient results due to the difficulties of training a model of hundreds of millions of parameters and, above all, the difficulty of capturing the long-term dependencies that occur in long text sequences. However, their performance has been tested in a task where the input does not exceed the maximum size of the pretrained model: the generation of synopses from book titles. An interesting potential of Transformers for text generation can be observed in this task, since the network generalizes the elaboration of a synopsis in such a way that it creates coherent arguments for book titles that do not exist; it is an artificial intelligence system that presents the capacity to generalize the rules of language as well as certain creativity; a Twitter bot has been enabled to disseminate its results. Finally, a new state of the art ROUGE summary metric is established on this corpus. Future work is proposed to improve the results.

Índice general

1. Introducción	1
1.1. Objetivos	2
1.2. Trabajos relacionados	3
1.3. Contribuciones	3
1.4. Estructuración	4
2. Estado del arte	5
2.1. Resumen automático	5
2.2. Modelos de distribución de palabras (Word Embeddings)	6
2.3. Modelado de lenguaje	9
2.4. Redes de Neuronas Recursivas	10
2.4.1. Neurona Recursiva	11
2.4.2. Long Short-Term Memory	12
2.5. Modelos codificador-descodificador	13
2.6. Modelos recursivos bidireccionales	14
2.7. Mecanismos de atención	16
2.7.1. Atención multiplicativa	18
2.7.2. Atención con perceptrón multicapa	18
2.8. Transformers	19
2.8.1. Arquitectura	20
2.8.2. Self Attention	20
2.8.3. Codificación posicional	24
2.8.4. MultiHead Attention	24
2.8.5. Descodificador	25
2.8.6. Capa de salida	26
2.9. BERT	27
2.9.1. Representación de la entrada	28
2.9.2. Arquitectura	29
2.9.3. Preentrenamiento	30
2.9.4. Ajuste fino	30

2.10. Sistema de evaluación de resumen	32
3. Estudio y descripción del conjunto de datos	35
4. Modelo propuesto	43
4.1. Arquitectura	43
4.1.1. Elección del codificador	43
4.2. Preparación de los datos para resumen	45
4.2.1. Segmentación de la entrada	45
4.3. Preparación de datos para generación de sinopsis	46
4.4. Implementación	47
5. Experimentos	49
5.1. Configuración	49
5.2. Conjunto de datos	50
5.3. Experimento 1 - Resumen abstractivo	51
5.4. Experimento 2 - Generación de sinopsis	53
6. Conclusiones y líneas futuras	57
A. Palabras frecuentes por categoría	59
Bibliografía	67

Índice de figuras

2.1. Word embeddings representadas en el espacio (Allen and Hospedales, 2019)	8
2.4. Modelo skip-Gram	8
2.2. Modelo CBOW	9
2.3. Intuición tras skip-gram	9
2.5. Representación interna de una neurona RNN simple sobre el tiempo	11
2.6. Arquitectura Encoder-Decoder simplificada	13
2.7. Arquitectura codificador-descodificador al detalle para un problema de traducción	15
2.8. Ejemplo de la bidireccionalidad de la secuencia lingüística.	15
2.9. Arquitectura de una red recursiva bidireccional	16
2.10. (Zhang et al., 2020a)	17
2.11. La salida de la capa de atención es una suma ponderada de los valores. (Zhang et al., 2020a)	18
2.12. Arquitectura general del Transformer (Vaswani et al., 2017)	19
2.13. Arquitectura general del Transformer en detalle (Alammar, 2019) .	21
2.14. Pesos de atención que relacionan el pronombre <i>it</i> con <i>animal</i> y <i>street</i> , capturando la posible ambigüedad semántica (Zhang et al., 2020a)	23
2.15. Codificación posicional (Alammar, 2019)	24
2.16. Tipos de <i>self-attention</i> utilizadas para entrenar una red neuronal. .	25
2.17. Atención en Transformers (Vaswani et al., 2017)	26
2.18. Las dos etapas del entrenamiento de BERT (Devlin et al., 2018) . .	27
2.19. Construcción de los vectores de entrada de BERT (Devlin et al., 2018)	29
2.20. Comparativa de BERT con otros modelos basados en Transformers (Devlin et al., 2018)	29
3.1. Número de libros por género	38
3.2. Libros con más de un género asignado	40
3.4. Distribución de la longitud de los libros	41

4.1. Arquitectura propuesta	44
4.2. Segmentación del texto	47

Índice de tablas

3.1. Forma del conjunto de datos: Título, Autor, Géneros, Colección	35
3.2. Forma del conjunto de datos: Sinopsis	36
3.3. Forma del conjunto de datos: Año, Páginas, Valoración, Nº votos	36
3.4. Descripción de los datos numéricos del conjunto	36
3.5. Descripción de los datos de texto del conjunto	37
3.6. Descripción de los datos del campo Idioma	37
3.7. Número de libros por género	39
3.8. Descripción de los campos del número de símbolos de cada libro y los resúmenes (unidades: símbolos)	40
4.1. Descripción de los distintos modelos de BERT (Devlin et al., 2018; Cañete et al., 2020)	44
5.1. Resumen de los parámetros utilizados	50
5.2. Experimento 1 - Resultados de ROUGE F1 para el modelo propuesto sobre el conjunto de datos de test para resumen abstractivo	52
5.3. Resultados de ROUGE F1 para varios modelos sobre el conjunto de datos de test CNN/Daily Mail. AVG es la media de ROUGE-1, ROUGE-2 y ROUGE-L. (Zhang et al., 2019a)	53
5.4. Experimento 1 - Producciones de resúmenes para libros célebres	53
5.5. Experimento 2 - Resultados de ROUGE F1 para el modelo propuesto sobre el conjunto de datos de test para generar sinopsis a partir del título	54
5.6. Experimento 2 - Producciones del modelo para algunos títulos	55

Capítulo 1

Introducción

La necesidad de generar resúmenes automáticos surge de manera natural cuando se dispone de una enorme fuente de información textual intratable manualmente. Internet, por ejemplo, es un gran texto; resumir su información de manera concisa ayudaría a la recolección y tratamiento de datos, así como a su visualización. Otro ejemplo son los libros: han sido el medio histórico de transmisión de texto. Dentro de la industria editorial, el resumen de textos literarios es de gran interés en el campo de la generación automática de informes de lectura. Un informe de lectura consta de clasificación de un manuscrito en un género determinado, inscripción en una tradición literaria (una especie de desambiguación de autor) y, por supuesto, un resumen, que es el elemento más costoso del informe. El interés comercial radica en que las editoriales reciben miles de manuscritos inéditos al año. La tarea de leer y crear informes de lectura la realizan los llamados lectores profesionales con un alto coste de tiempo y recursos. Diseñar una herramienta o modelo lingüístico basado en inteligencia artificial capaz de generar resúmenes a partir de manuscritos, mejoraría la productividad y eficacia de estos lectores. Por todo ello, la generación automática de resumen de textos literarios es el objetivo de este trabajo.

Sin embargo, cuando se aborda el problema de resumen abstractivo en español, se encuentran dos grandes dificultades. La primera y más importante es que no existe un conjunto de datos en español donde se puedan encontrar libros asociados con el resumen abstractivo escrito por humanos. La segunda es que, de existir, el tamaño del conjunto de datos y las secuencias de entrada serían de una magnitud intratable de manera inmediata, dadas las dificultades de los modelos de aprendizaje profundo para capturar las dependencias a largo plazo que se dan en un libro (Hochreiter et al., 2001). Por lo tanto, surgen las necesidades de la fabricación de un conjunto de datos para resumen abstractivo y la búsqueda de una arquitectura que pueda manejar tales entradas en un tiempo de entrena-

miento asumible para un trabajo académico de estas características.

Afortunadamente, para el problema del conjunto de datos, gracias a la paulatina digitalización de libros, las editoriales ofrecen un catálogo mayor de textos asociados con determinados metadatos como su autor, fecha de publicación, la valoración de los usuarios, y, más importante, su sinopsis. Este proyecto ha elaborado, por medio de técnicas de *scrapping*, a partir de una biblioteca de más de 40 mil libros electrónicos, un conjunto de datos de libros etiquetados por su autor, sinopsis, fecha de publicación, valoración de los usuarios, entre otros campos.

No obstante, resta la dificultad del entrenamiento de un modelo de aprendizaje automático sobre una gran cantidad de datos. La transferencia de conocimiento (o *transfer learning*) (Pan and Yang, 2009), ha mejorado los resultados de un número significativo de tareas de aprendizaje automático, aplicando el conocimiento adquirido por el entrenamiento de una tarea más general hacia otra más específica. Es el caso del modelo "Bidirectional Encoder Representations from Transformers" (BERT) (Devlin et al., 2018), que ha batido varias marcas en determinadas pruebas de rendimiento de Procesamiento de Lenguaje Natural como SQuAD (Rajpurkar et al., 2016) y GLUE (Wang et al., 2018). Este tipo de modelos, en una primera etapa de preentrenamiento, adquieren nociones lingüísticas fundamentales, que posteriormente se utilizan para un ajuste fino en tareas específicas como la clasificación, traducción o resumen. Así, frente a la gran cantidad de tiempo que consume entrenar un modelo sobre un corpus de gran tamaño, solamente se requieren unas horas o pocos días de entrenamiento para ajustarlo a una tarea específica. Esto, añadido a que se basan en la simulación de mecanismos cognitivos de atención, muy semejantes a los que utilizamos los humanos en el proceso de lectura, motiva la pregunta: ¿cuál es el rendimiento de los de modelos lingüísticos *Transformers* en la generación de resumen de textos literarios en español?

Este trabajo plantea una primera aproximación a este problema creando un conjunto de textos literarios etiquetados y entrenando una arquitectura para resumen basada en BERT. Como tarea menos ambiciosa pero interesante, se prueba también, con el mismo modelo, la generación automática de sinopsis a partir de un título.

1.1. Objetivos

1. Construir un corpus etiquetado de textos literarios en español, dando prioridad al texto y su sinopsis.
2. Describir el conjunto de datos para esta y otras posibles investigaciones fu-

turas.

3. Estudiar los modelos lingüísticos Transformers y sus variantes preentrenado. Escoger la implementación de uno de ellos.
4. Entrenar sobre el conjunto de datos el modelo escogido.
5. Evaluar los resultados cualitativa y cuantitativamente.

1.2. Trabajos relacionados

Los trabajos más relevantes para resumen abstractivo con BERT son los de Zhang et al. (2019a) y Liu and Lapata (2019). Levemente relacionado está Zhang et al. (2020b) que presenta un modelo para comprensión documentos largos. Sin el empleo de BERT, existen trabajos como Raffel et al. (2019), quienes entrena un sistema multiobjetivo siendo uno de ellos el resumen. También se encuentra, en resumen extractivo, distinto al que nos ocupa, los trabajos Zhang et al. (2019b) y Liu (2019).

1.3. Contribuciones

Este trabajo presenta las siguientes contribuciones:

- La elaboración y descripción de un conjunto de datos etiquetado de textos literarios en español, para su utilización en problemas de Procesamiento de Lenguaje Natural como resumen, clasificación, desambiguación de autor o datación automática de textos.
- La investigación bibliográfica de la Sección 2.9.4, donde se tratan los problemas de los modelos BERT preentrenados para textos de gran longitud.
- Propuesta de una arquitectura para resumen abstractivo utilizando BERT.
- El entrenamiento de dicho modelo sobre textos en español para resumen abstractivo.
- El entrenamiento del modelo para la generación de sinopsis a partir de los títulos.
- La demostración de que el nivel del estado del arte en una lengua distinta al inglés dista de ofrecer resultados semejantes para resumen.

- Resultados que demuestran las limitaciones de los modelos Transformer para textos de longitud superior a la que han sido preentrenados.
- Para tareas cuya entrada requiere un tamaño igual o inferior, se aporta una red que generaliza la elaboración de una sinopsis de tal manera que crea argumentos coherentes para títulos de libros que no existen, lo que supone la demostración de su capacidad de generalización de las reglas del lenguaje, así como cierta creatividad.
- Sugerencias de trabajos futuros para tratar de solventar estas limitaciones.
- Al ser un conjunto de datos nuevo, se establece un estado del arte base para las métricas ROUGE (Lin, 2005).

1.4. Estructuración

El trabajo se estructura en tres grandes bloques: los fundamentos teóricos del Capítulo 2, el diseño y estudio del conjunto de datos (Capítulo 3) y la implementación, entrenamiento y evaluación de un modelo basado en BERT para resumen abstractivo y generación de sinopsis (Capítulos 4 y 5). Por último, en el Capítulo 6 se resume el contenido y se exponen las conclusiones.

Capítulo 2

Estado del arte

En este capítulo se presenta, en primer lugar, una definición de la tarea de resumen y sus tipos. En segundo, se exponen los principios teóricos en los que se basa el modelado de lenguaje con Redes de Neuronas Artificiales, desde las Redes de Neuronas Recursivas (Elman, 1990) hasta los últimos modelos Transformers (Vaswani et al., 2017) y sus variantes preentrenadas (Devlin et al., 2018; Dai et al., 2019). Se discuten las ventajas e inconvenientes de cada arquitectura. La discusión sobre los inconvenientes de BERT para resumen, realizada en la Sección 2.9.4, es una de las novedades de este trabajo.

Al mismo tiempo que se desarrolla la teoría, se orientan las decisiones de modelado que se tomarán en el Capítulo 4. Para su comprensión adecuada, se asumen conocimientos del comportamiento de Redes de Neuronas Alimentadas hacia delante (Hornik et al., 1989), regresión logística (Kleinbaum et al., 2002) y del algoritmo de BackPropagation (Rumelhart et al., 1986).

2.1. Resumen automático

Un resumen es un texto corto generado a partir de otro de mayor longitud sin modificar su información sustancial. (Nenkova and McKeown, 2011).

Existen distintos tipos de resumen:

- *Short Tail Summarization* (resumen de textos cortos). Son resúmenes sobre textos de corta longitud. Por ello, el resumen resultante ha de tener la precisión suficiente como para no cambiar el significado del texto.
- *Long Tail Summarization*: Son resúmenes sobre un contenido de gran longitud tal que un ser humano tendría dificultades para llevarlo a cabo. Se trata de un resumen de cientos de páginas o libros enteros.

- Generación de títulos: un tipo especial de resumen, donde los títulos encapsulan y sugieren el contenido del libro.
- Resumen de palabras clave: se trata de extraer aquellas palabras que son relevantes en el texto, de modo que, conjuntamente, den una idea de su contenido.

Para conseguir cada uno de estos tipos de resumen, se encuentran dos grandes grupos de técnicas:

- Técnicas de resumen extractivo: Consiste en la extracción de las frases más importantes del contenido, reunidas para obtener un resumen simple y corto. Intuitivamente se interpreta como un *subrayado* del texto.
- Técnicas de resumen abstractivo: consiste en la generación de frases y palabras que no necesariamente se encuentran en el texto pero mantienen y comprimen el significado del contenido. Conlleva una *reescritura* de la información. Es el tipo de técnica de resumen que utilizamos los humanos.

2.2. Modelos de distribución de palabras (Word Embeddings)

Las palabras son un tipo de dato discreto. Sin embargo, la entrada de una red de neuronas artificial requiere un vector de entrada numérico y continuo.

Tradicionalmente, para crear una representación vectorial de una palabra, se utilizaban los vectores codificados con *one-hot*. Esta codificación consiste en que el vector que representa la palabra tiene el mismo tamaño que el vocabulario y la aparición de ese símbolo se representa con un 1, dejando a 0 el resto.

La utilización de este tipo de representación en Aprendizaje Automático puede llevar a la *maldición de la dimensión* o efecto Hughes, que hace difícil, lento o imposible el aprendizaje de la red (Bengio and Bengio, 2000).

Las *word embeddings* tratan de convertir los símbolos discretos en un punto de un espacio vectorial, de manera que se reduzca la dimensión del espacio del problema y se capturen relaciones de significado.

Con las representaciones distribuidas, las palabras quedan caracterizadas en un vector de dimensiones más bajas (por ejemplo, d=100, en contraposición al tamaño de todo el vocabulario, que puede ser alrededor de 100 órdenes mayor). El significado y otras propiedades de la palabra quedan representadas a través de las diferentes dimensiones de este vector.

2.2. MODELOS DE DISTRIBUCIÓN DE PALABRAS (WORD EMBEDDINGS) 7

Además de ser una solución a la *maldición de la dimensión*, las *word embeddings* ofrecen varios beneficios:

1. La reducción de la dimensión del vector de entrada es eficiente desde el punto de vista computacional.
2. Las representaciones basadas en la frecuencia de las palabras, como TF-IDF (Ramos et al., 2003) dan como resultado vectores de dimensiones altas que codifican información de manera redundante a lo largo de varias dimensiones. Además, no establecen relaciones semánticas.
3. Las representaciones de vectores de palabras aprendidos a partir de los datos específicos de una tarea (por ejemplo, la tarea de resumen), suelen ser óptimos para la tarea en cuestión.

Entrenamiento de las word embeddings Todos los métodos de entrenamiento de *word embeddings* utilizan como conjunto de datos sólo palabras (es decir, datos sin etiquetar), pero se entrena de forma supervisada. Esto es posible porque se diseñan tareas auxiliares supervisadas en las que los datos están etiquetados implícitamente. Las tareas auxiliares típicas son:

- Dada una secuencia de palabras, se predice la siguiente palabra. Es la denominada tarea de *modelado de lenguaje*. Por su importancia, se profundizará sobre ella en la Sección 2.3.
- Dada una secuencia de palabras antes y después de una posición, se predice la palabra de esa posición.
- Dada una palabra, se predicen las que ocurren dentro de una ventana (contexto), independientemente de la posición.

Este tipo de representaciones permitieron mejorar el rendimiento de los modelos en muchas tareas de Procesamiento de Lenguaje Natural cuando se integraron con las Redes de Neuronas Artificiales (Zou et al., 2013; Weiss et al., 2015; Kim, 2014).

Estos mecanismos de representación de palabras se han popularizado gracias a Word2vec (Mikolov et al., 2013). Dos de los más populares son Skip-Gram y CBOW:

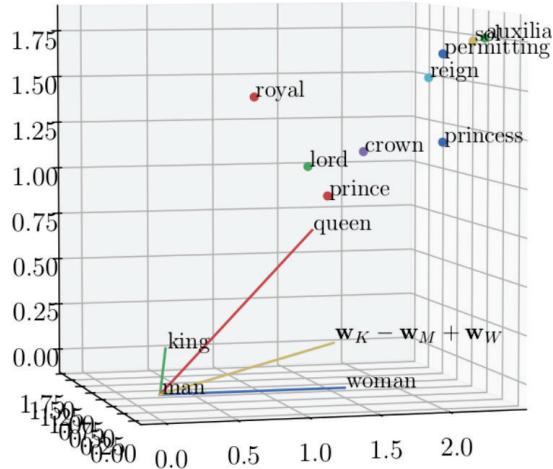


Figura 2.1: Word embeddings representadas en el espacio (Allen and Hospedales, 2019)

Continuous Bag of Words (CBOW) CBOW se basa en la probabilidad de aparición de una palabra condicionada por un contexto de tamaño k . Como se muestra en la Figura 2.2 el modelo CBOW es una red neuronal con una capa oculta. En la capa de entrada se introduce el vector de la palabra-contexto. La capa de salida es la probabilidad *softmax* sobre el vocabulario. El objetivo del modelo es predecir una palabra dado su contexto. Para un vocabulario de tamaño V , y una capa oculta de tamaño N . Las capas adyacentes están totalmente conectadas. La entrada está codificada con *one-hot vector*.

Skip-Gram El modelo de Skip-gram hace exactamente lo contrario del modelo CBOW: predice las palabras que aparecen en el contexto de una palabra (Figura 2.3). En la Figura 2.4 se contempla un resumen de su arquitectura.

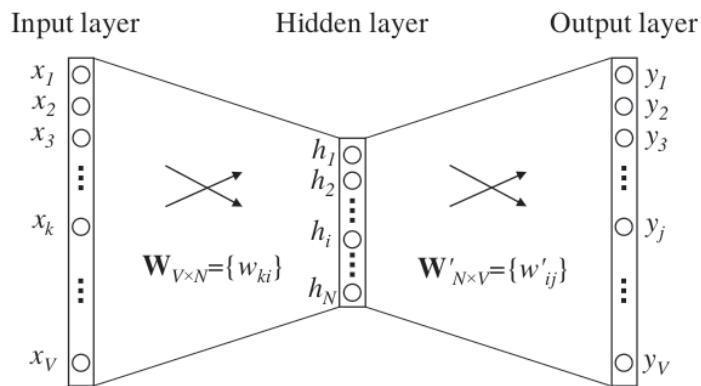


Figura 2.4: Modelo skip-Gram (Rong, 2014)

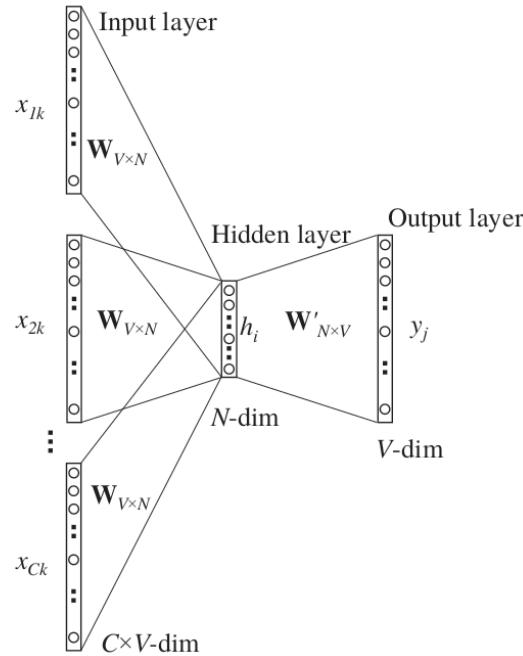


Figura 2.2: Modelo CBOW (Rong, 2014)

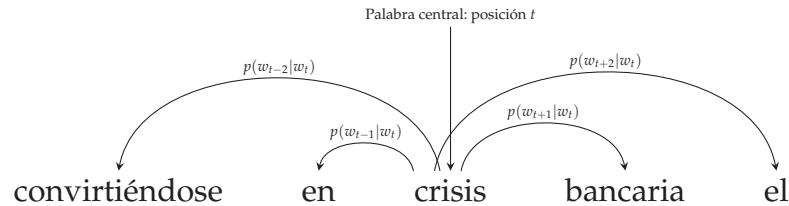


Figura 2.3: Intuición tras skip-gram

2.3. Modelado de lenguaje

Un modelo lingüístico (ML) consiste en una distribución de probabilidad sobre secuencias de símbolos $y = y_1, \dots, y_m$:

$$p(y) = \prod_{i=1}^m p(y_i | y_{<i}) \quad (2.1)$$

En la Ecuación 2.1, cada uno de los términos individuales $p(y_i | y_{<i})$ es la probabilidad condicionada de una palabra en este instante y_i , dados los símbolos anteriores $y_{<i}$. Se puede interpretar como una idea de *contexto*. Tradicionalmente, para modelar esta probabilidad condicionada, los modelos lingüísticos basados en n -gramas han recurrido a la hipótesis de Markov que modela el contexto con

una ventana de tamaño fijo de $n - 1$ palabras, $p(y_i|y_{i-n+1}, \dots, y_{i-1})$. De hecho, los modelos lingüísticos basados en n -gramas han de almacenar y procesar todos los n -gramas posibles que puedan ocurrir en el corpus de entrenamiento; para determinados problemas esta hipótesis era inviable y por ello se propusieron las Redes de Neuronas Recursivas.

2.4. Redes de Neuronas Recursivas

Además de un tipo de dato discreto (Sección 2.2), el lenguaje es una información secuencial; es decir, el orden en el que se sitúan las palabras en la cadena textual influye en el significado. Dicho de otro modo, las palabras tienen un significado u otro en función de las palabras anteriores. Un ejemplo sería la diferencia entre «perrito» y «perrito caliente».

Las Redes de Neuronas Recursivas (RNN) están hechas a medida para modelar tales dependencias de contexto en el lenguaje. El término *recursivo* se refiere a la idea de operar sobre cada entrada secuencialmente, de manera que la salida depende de los resultados de las operaciones previas. Generalmente, en Procesamiento de Lenguaje Natural, la red tiene como entrada un vector de tamaño fijo y se alimenta la red símbolo a símbolo. Las unidades (*símbolos*) son caracteres, palabras o incluso oraciones. Surgieron porque las Redes de Neuronas Alimentadas hacia delante no son capaces de capturar la dependencia entre datos alimentados secuencialmente en la red.

En la Sección 2.3, se ha introducido brevemente un modelo n -grama de Markov, donde si se quiere calcular la probabilidad de que una palabra aparezca en un determinado contexto se ha de escoger un tamaño de ventana n , de manera que el número de parámetros del modelo se incrementa exponencialmente, ya que hay que almacenar $|V|^n$ valores para un vocabulario V . Por ello, en lugar de modelar el lenguaje como $p(x_t|x_{t-1}, \dots, x_{t-n+1})$, las Redes de Neuronas Recursivas utilizan un modelo de estados ocultos:

$$p(x_t|x_{t-1}, \dots, x_1) \approx p(x_t|x_{t-1}, h_t) \quad (2.2)$$

donde h_t es el estado oculto propiamente dicho, que almacena una representación de la secuencia.

En otras palabras, las Redes de Neuronas Recursivas *memorizan* el resultado de los cálculos de instantes anteriores y utilizan esta información latente para el cálculo que se ejecuta en un determinado instante. Agregar memoria a las redes neuronales tiene un propósito: hay información en la secuencia misma, en la posición de cada símbolo, y las redes recursivas la usan para realizar tareas que

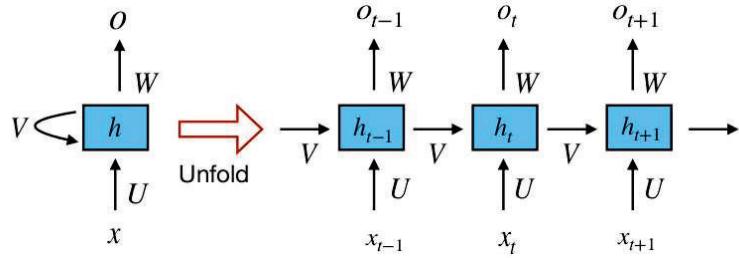


Figura 2.5: Representación interna de una neurona RNN simple sobre el tiempo (LeCun et al., 2015)

el perceptrón multicapa no puede: las RNN tienen la capacidad de capturar la naturaleza secuencial inherente en el lenguaje.

Otra ventaja es que las RNN permiten modelar tareas secuenciales cuya entrada es de longitud variable. Esto ofrece una flexibilidad que dota a las RNN de una mejor capacidad de modelización y la posibilidad de capturar el contexto semántico sin límites de longitud.

Muchas tareas de Procesamiento de Lenguaje Natural requieren un modelado semántico o *codificación* de toda la frase. Ello implica crear una representación semántica latente de la oración en un espacio dimensional fijo, un vector. La capacidad de las RNN para comprimir frases a un vector llevó a su popularización para tareas de traducción automática (Cho et al., 2014), en la que toda la frase se resume en un vector fijo y luego descodifica la secuencia de destino de longitud variable. Aunque las RNN no se utilizan en este trabajo. Todas las intuiciones de la Sección 2.8 sobre Transformers se asientan en la evolución natural de estas ideas. Por ello se considera necesaria su explicación.

2.4.1. Neurona Recursiva

Elman (1990) introdujo la red en la que se basan los modelos de RNN; originalmente se trataba de una red de tres capas. Como se puede ver en la figura 2.5, x_t es la entrada de la red en el instante t y h_t representa el estado oculto en ese instante. h_t se calcula del siguiente modo:

$$h_t = f(Ux_t + Wh_{t-1}) \quad (2.3)$$

h_t se calcula basándose en la entrada actual y en el estado oculto del paso de tiempo anterior. La función f es una función no lineal como tanh o ReLU (Xu et al., 2015). U , V , W son las matrices de pesos que se comparten a lo largo del tiempo.

En el contexto de Procesamiento de Lenguaje Natural, x_t suelen ser un vec-

tor de distribución de palabras (*word embeddings*) o un *one-hot* vector. A veces, también pueden ser representaciones abstractas del contenido textual, como, por ejemplo, producciones de una gramática.

o_t es la salida de la red, que se pasa por una función no lineal. Como ya se ha dicho, se puede considerar como el elemento de memoria de la red que acumula información de otras etapas temporales.

En la práctica, sin embargo, estas redes RNN simples padecen el problema del *desvanecimiento del gradiente* (Hochreiter and Schmidhuber, 1997)¹, lo que dificulta mucho el aprendizaje y la elección de los hiperparámetros de la red. Para superar esta limitación se propusieron las neuronas de memoria a corto plazo (*Long Short-Term Memory*, LSTM).

2.4.2. Long Short-Term Memory

Las neuronas *Long Short-Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997) añaden una puerta adicional que se encarga de *olvidar*. Se propuso como alternativa para solucionar el problema del desvanecimiento del gradiente.

En las Redes Recursivas sencillas, el error no se propaga correctamente en un ilimitado número de instantes t . Las LSTM permiten esto gracias a tres puertas: una puerta de entrada, una que se encarga de *olvidar* y otra de salida. El estado oculto de la neurona se calcula teniendo en cuenta el resultado que arrojan estas tres puertas de la neurona. Las ecuaciones que representan los estados internos de las neuronas son los siguientes:

$$x = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \quad (2.4)$$

$$f_t = \sigma(W_f \cdot x + b_f) \quad (2.5)$$

$$i_t = \sigma(W_i \cdot x + b_i) \quad (2.6)$$

$$o_t = \sigma(W_o \cdot x + b_o) \quad (2.7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot x + b_c) \quad (2.8)$$

¹A medida que se añaden más capas que utilizan ciertas funciones de activación como la sigmoidal, los gradientes de la función de pérdida se acercan a cero, lo que hace que la red sea difícil de entrenar.

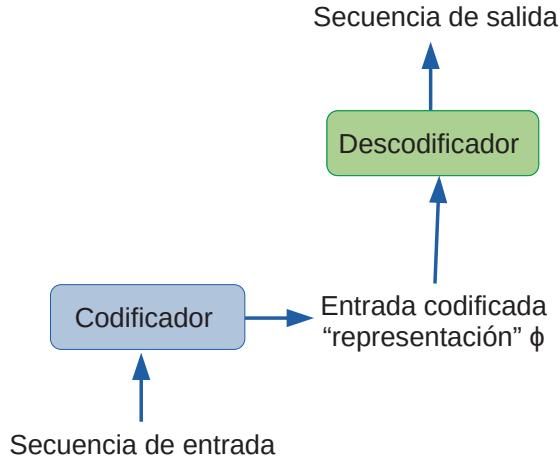


Figura 2.6: Arquitectura Encoder-Decoder simplificada

$$h_t = o_t \odot \tanh(c_t) \quad (2.9)$$

Los valores iniciales son $c_0 = 0$ y $h_0 = 0$ y el operador \odot denota el producto de Hadamard². El subíndice t es el índice del instante de tiempo.

Las variables son $x_t \in \mathbb{R}^d$, $f_t \in \mathbb{R}^h$, $i_t \in \mathbb{R}^h$, $o_t \in \mathbb{R}^h$, $h_t \in \mathbb{R}^h$, $c_t \in \mathbb{R}^h$, $W \in \mathbb{R}^{h \times d}$ σ es la función de activación sigmoidal.

2.5. Modelos codificador-descodificador

La arquitectura codificador-descodificador es una composición de dos modelos (Cho et al., 2014): uno que se encarga de codificar la entrada y otro que, a partir de dicha codificación, descodifica una secuencia objetivo; se suelen entrenar en conjunto (Figura 2.6). Formalmente, el codificador produce una representación interna ϕ de la entrada en un vector. El objetivo del descodificador es tomar la entrada codificada y producir la salida deseada. Tanto las entradas y como las salidas son secuencias, a menudo de longitudes diferentes; por ello, también se

²El producto de Hadamard es una operación binaria que toma dos matrices de las mismas dimensiones y produce otra matriz de la misma dimensión que los operandos donde cada elemento i, j es el producto de los elementos i, j de las dos matrices originales.

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}$$

suelen denominar modelos "Secuencia-a-Secuencia" (Seq2Seq).

Se puede aplicar a multitud de problemas. Por ejemplo, para resumen, la secuencia de entrada sería el texto de un libro y la secuencia de salida el resumen en sí. En la traducción automática, la entrada sería el texto para traducir y la salida, el texto traducido.

Una forma de ver los modelos de codificador-decodificador es como un caso especial de los llamados *modelos de generación condicionada*. En la generación condicionada, en lugar de la representación de entrada ϕ , un contexto general de condicionamiento c influye en el decodificador para producir una salida. No todos los modelos de generación condicionada tienen un codificador, pues es posible que el contexto de condicionamiento se derive de otra fuente. Por ejemplo, en un sistema generador de informes meteorológicos, los valores de la temperatura, la humedad y la velocidad y dirección del viento podrían "condicionar" un decodificador para generar el informe meteorológico.

En Aprendizaje Profundo se han empleado Redes Recursivas (Sutskever et al., 2014) como codificador y decodificador (Figura 2.7).

Estos modelos han sido ampliados con dos ideas fundamentales. La primera es la representación bidireccional que combina en el entrenamiento la propagación hacia adelante y hacia atrás sobre la secuencia para crear representaciones más ricas. La segunda idea han sido los mecanismos de atención, muy útiles para que el modelo se centre en las partes de la entrada que son relevantes. A continuación, se ahondará en estas dos ideas.

2.6. Modelos recursivos bidireccionales

Una forma de entender un modelo recursivo es verlo como una caja negra que codifica una secuencia en un vector. Cuando se modela una secuencia, es útil observar no sólo las palabras del pasado sino también las que aparecen en el futuro. Por ejemplo, como puede verse en la secuencia de la Figura 2.8, para poder resolver la referencia anafórica del pronombre "lo" es necesario leer "hacia atrás".

En conjunto, la información del pasado y del futuro permite representar con solidez el significado de una palabra en una secuencia. Este es el objetivo de los modelos bidireccionales recursivos (Schuster and Paliwal, 1997) y cuyo modelo más sofisticado y popular es ELMO (Peters et al., 2018). En la Figura 2.9 puede contemplarse la arquitectura típica, donde dos modelos RNN codifican la secuencia en ambas direcciones y producen un vector resultante concatenando las dos representaciones.

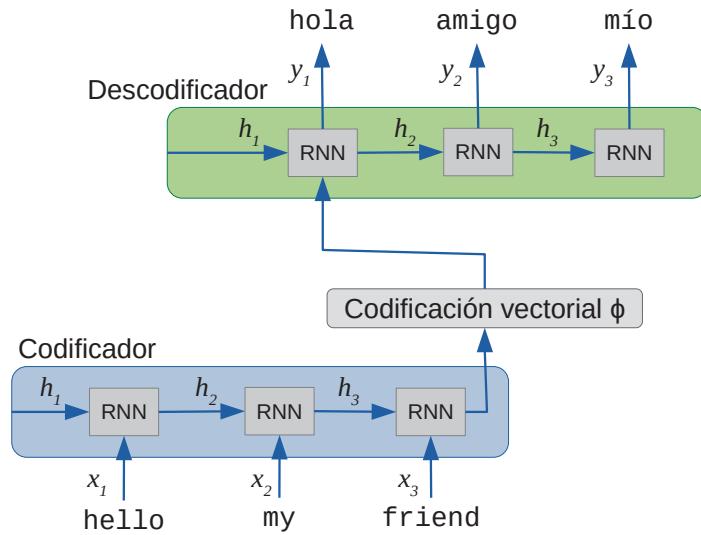


Figura 2.7: Arquitectura codificador-descodificador al detalle para un problema de traducción

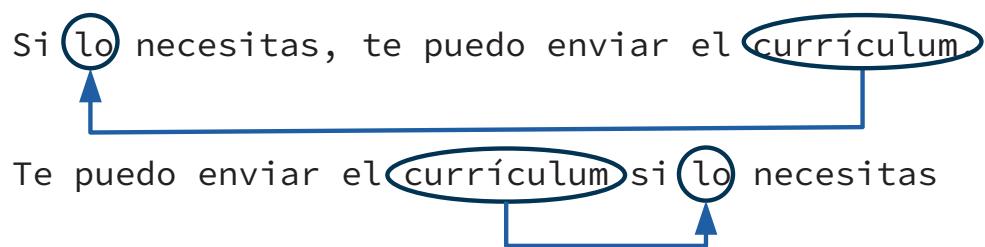


Figura 2.8: Ejemplo de la bidireccionalidad de la secuencia lingüística. Para comprender el texto de la primera frase es imprescindible tener una representación derecha-izquierda

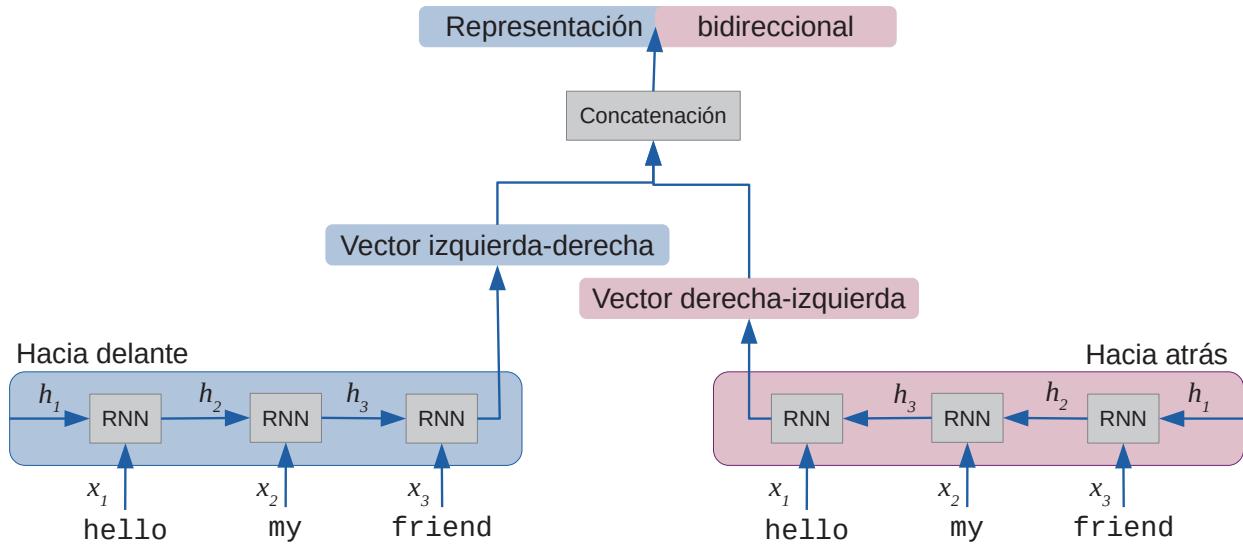


Figura 2.9: Arquitectura de una red recursiva bidireccional

2.7. Mecanismos de atención

La concentración permite a los humanos priorizar la percepción de unos elementos sobre otros; no procesamos toda la información que está disponible ante nuestros sentidos, sino que prestamos atención a una pequeña parte del entorno. En la neurociencia cognitiva, hay varios tipos de atención como la atención selectiva o la atención espacial. La aplicación de este tipo de mecanismos cognitivos al Aprendizaje Profundo se basa en la teoría de integración de características de la atención selectiva de Treisman and Gelade (1980). Esta Sección se centrará en la aplicación de esta idea al aprendizaje profundo, donde la atención puede interpretarse, de manera simplificada, como un método generalizado de alineación de las entradas sobre las salidas de la red.

En la Sección 2.5, el codificador convertía la secuencia de entrada en un estado oculto que posteriormente se introducía en el decodificador para generar la secuencia objetivo. En tareas como la traducción o el resumen, un símbolo de entrada puede estar estrechamente relacionado con un símbolo de salida. Por ejemplo, si se traduce "Hello word" a "Bonjour le monde", "hello" está totalmente alineado con "bonjour", así como "world" con "monde". En los modelos Seq2Seq que utilizan redes LSTM como el de la Figura 2.7, el decodificador aprende implícitamente a olvidar las partes de la secuencia que no tienen relación entre sí. Los mecanismos de atención tratan de hacer este tipo de selecciones explícitas.

En la Figura 2.10 se observa una capa de atención. La entrada de la capa de atención se denomina *query* o consulta. Para cada consulta, la capa devuelve una

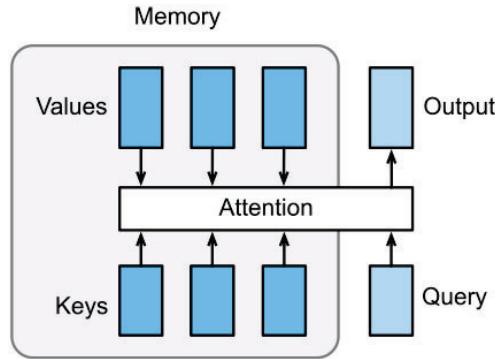


Figura 2.10: (Zhang et al., 2020a)

salida utilizando su memoria. La memoria es un conjunto de pares clave-valor (*keys-values*). Formalmente, la memoria son n pares clave-valor, $(\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_n, \mathbf{v}_n)$ con $\mathbf{k}_i \in \mathbb{R}^{d_k}, \mathbf{v}_i \in \mathbb{R}^{d_v}$. Dada una consulta, $\mathbf{q} \in \mathbb{R}^{d_q}$, la capa de atención devuelve una salida $\mathbf{o} \in \mathbb{R}^{d_v}$ con la misma forma del vector de valores.

El proceso completo del mecanismo de atención se puede observar en la Figura 2.11. Para calcular la salida de la capa de atención, se utiliza la función α , que mide la similitud entre la consulta y la clave. Intuitivamente, la función α indica cuánta atención se ha de prestar a cada elemento de la entrada respecto a su salida.

Para cada clave $(\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_n, \mathbf{v}_n)$, se calcula la puntuación:

$$a_i = \alpha(\mathbf{q}, \mathbf{k}_i) \quad (2.10)$$

A continuación, se utiliza *softmax* para obtener los pesos de atención:

$$\mathbf{b} = \text{softmax}(\mathbf{a}), \text{ donde } b_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)}, \mathbf{b} = [b_1, \dots, b_n]^T \quad (2.11)$$

La suma se pondera para obtener los pesos de salida (Figura 2.11):

$$\mathbf{o} = \sum_{i=1}^n b_i \mathbf{v}_i \quad (2.12)$$

Las diferentes funciones de puntuación dan lugar a los distintos tipos de capas de atención existentes. Las más comunes son la atención multiplicativa y atención con un perceptrón multicapa.

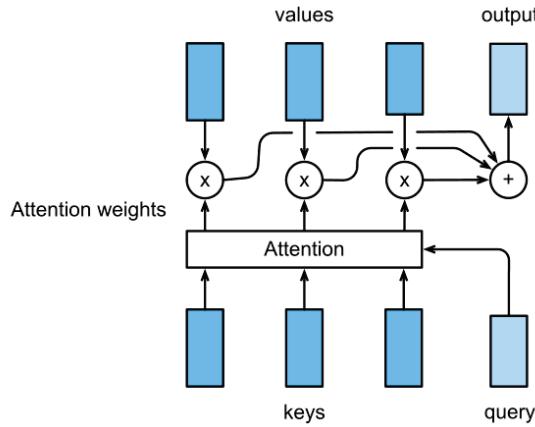


Figura 2.11: La salida de la capa de atención es una suma ponderada de los valores. (Zhang et al., 2020a)

2.7.1. Atención multiplicativa

Se asume que el vector de la consulta tiene la misma dimensión que las claves, denominadas $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$ para cada i . El producto calcula las puntuaciones multiplicando la consulta con las claves, y se divide entre \sqrt{d} para minimizar la influencia de la dimensión d en las puntuaciones:

$$\alpha(\mathbf{q}, \mathbf{k}) = \langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d} \quad (2.13)$$

Esta operación se puede generalizar para multiplicar consultas y claves de distintas dimensiones. Se asume que $\mathbf{Q} \in \mathbb{R}^{m \times d}$ contiene m consultas y $\mathbf{K} \in \mathbb{R}^{n \times d}$ contiene todas las n claves. Se calculan todas las puntuaciones mn :

$$\alpha(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^\top / \sqrt{d} \quad (2.14)$$

2.7.2. Atención con perceptrón multicapa

Con este mecanismo, se proyectan tanto las claves como las consultas en \mathbb{R}^h utilizando pesos aprendidos. Se asumen que los pesos a aprender son $\mathbf{W}_k \in \mathbb{R}^{h \times d_k}, \mathbf{W}_q \in \mathbb{R}^{h \times d_q}$ y $\mathbf{v} \in \mathbb{R}^h$. Así la función de puntuación queda definida por:

$$\alpha(\mathbf{k}, \mathbf{q}) = \mathbf{v}^\top \tanh (\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q}) \quad (2.15)$$

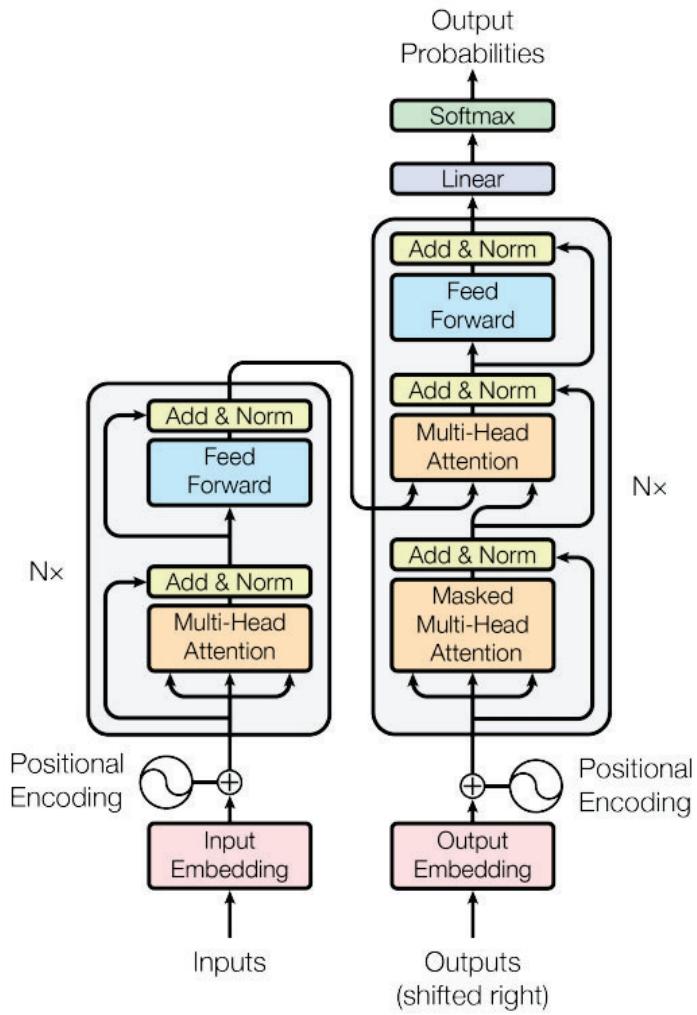


Figura 2.12: Arquitectura general del Transformer (Vaswani et al., 2017)

2.8. Transformers

En la Sección 2.4 se expusieron las Redes de Neuronas Recursivas. El principal inconveniente de estas arquitecturas es que su entrenamiento es muy lento, porque ha de ser secuencial y no permite fácilmente la transferencia de los pesos del modelo. El mayor beneficio de Transformer es la paralelización del aprendizaje.

El Transformer (Vaswani et al., 2017), igual que los modelos Seq2Seq, consiste en dos grandes bloques: un codificador y un decodificador, es decir, utiliza la arquitectura encoder-decoder, con una composición interna novedosa que prescinde de las Neuronas Recursivas (Figura 2.12).

2.8.1. Arquitectura

Encoder El bloque codificador se compone, a su vez, de seis codificadores apilados. Cada uno se subdivide en dos subcapas. La primera es una capa *multi-head attention* (con unidades *self-attention*), que será explicada en la Sección 2.8.2. La segunda, una red alimentada hacia delante. Se puede visualizar en la parte izquierda de la Figura 2.13. Ambas subcapas tienen conexiones residuales (He et al., 2016), y a continuación están conectadas a una capa de normalización (Ba et al., 2016).

Las conexiones residuales tienen la finalidad de evitar el problema del desvanecimiento del gradiente mediante la reutilización de los pesos de una capa anterior mientras la capa adyacente aprende sus pesos. Además, las conexiones residuales simplifican la red, ya que permiten utilizar menos capas en los pasos iniciales del entrenamiento. Esto acelera el aprendizaje, ya que hay menos capas por las que propagarse y evita el desvanecimiento del gradiente. La red restaura gradualmente las capas residuales a medida que aprende el espacio de características.

La normalización se utiliza para reducir el tiempo de entrenamiento. Se calcula la media y la varianza de la distribución de los datos de un minilote y posteriormente se utilizan para normalizar la entrada de la capa siguiente. Esto reduce significativamente el tiempo de convergencia del aprendizaje en las redes neuronales alimentadas hacia delante.

Decoder El descodificador, además de las subcapas del codificador, se le añade una tercera después de la de *self-attention*. Se trata de añadir otra capa de atención de manera que se atienda tanto a los estados internos del descodificador como a la salida de codificador. En este bloque también se aplica una conexión residual y la normalización de capas después de cada capa (parte derecha de la Figura 2.13)

2.8.2. Self Attention

El mecanismo de *self-attention* es semejante a la atención multiplicativa (*dot-product attention* en inglés) ya explicada en la Sección 2.7.1. Se compone de un vector consulta, clave y valor. La capa *self-attention* produce una salida secuencial de la misma longitud para cada elemento de entrada. En comparación con las Redes de Neuronas Recursivas, los elementos de salida de una capa de *self-attention* se pueden calcular en paralelo y, por lo tanto, se obtiene una implementación altamente eficiente.

Las operaciones de la capa de *self-attention* son las siguientes:

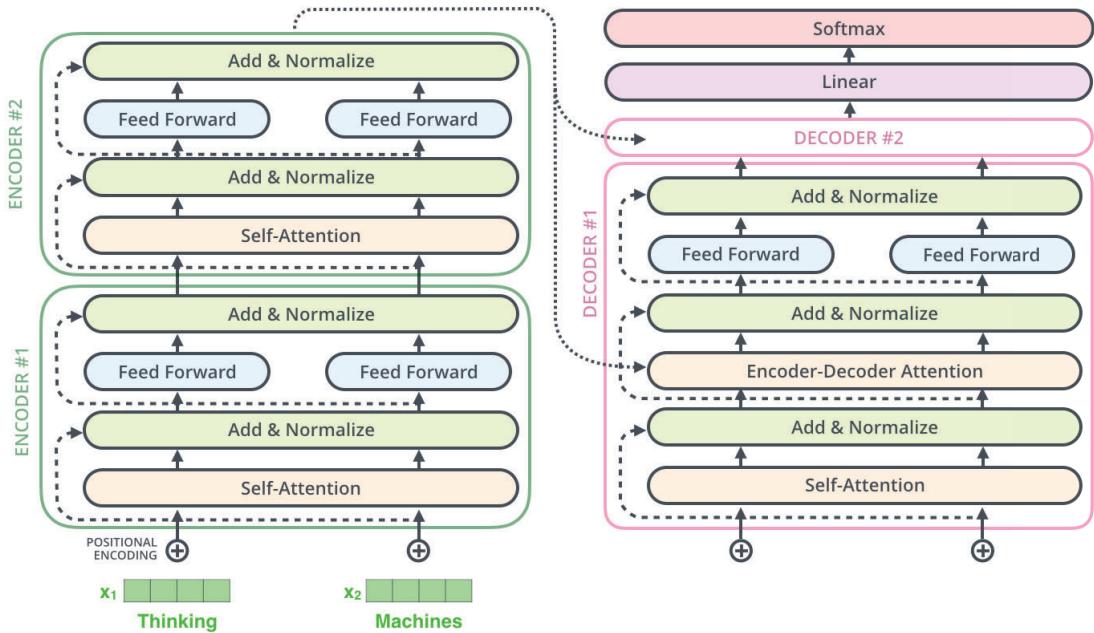


Figura 2.13: Arquitectura general del Transformer en detalle (Alammar, 2019)

1. **Calcular Q, K, V :** cada representación de palabras de entrada se utiliza con unos parámetros entrenables W^Q, W^K y W^V para calcular el vector de la consulta, la clave y el valor. Se obtienen multiplicando el vector de entrada con la matriz correspondiente.
2. **Puntuar similitud:** se calcula el valor de la atención. Por ejemplo, para la frase "Yo veo una película en la tablet y luego la termino en la televisión.". La puntuación de atención para la palabra de entrada "Yo", correspondiente a su consulta q_1 , se calcula multiplicando este vector con cada una de las claves k_i de la secuencia. Posteriormente, se realizan técnicas de normalización, y por medio de *softmax* se escalan de forma no lineal los valores de los pesos entre 0 y 1. Esto se debe a que el producto devuelve magnitudes muy grandes con dimensiones vectoriales también grandes (d), que resultarán en gradientes muy pequeños al pasar a la función *softmax*, se escalan los valores mediante $escala = 1/\sqrt{d}$
3. **Suma de los pesos** para producir la salida final, se suman los pesos *softmax* con el valor escogido.

Este proceso queda resumido en la Ecuación 2.16

$$\begin{aligned} X \times W^Q &= Q \\ X \times W^K &= K \quad \text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) V = Z \\ X \times W^V &= V \end{aligned} \quad (2.16)$$

Como se observa en la Ecuación anterior, el proceso es semejante sino idéntico a la atención multiplicativa (Sección 2.7.1). Sin embargo, en la Figura 2.12 se advierten tres bloques de atención separados: (i) el *Encoder Attention*, (ii) el *Decoder Attention* y (iii) el *Encoder-Decoder Attention*. Cada uno difiere en la naturaleza de sus vectores $Q/K/V$:

- **Bloque Encoder Attention**

Q = el vector de la palabra de entrada actual con información posicional

K = todos los vectores de palabras con información posicional en la secuencia de entrada

V = todos los vectores de palabras con información posicional en la secuencia de entrada

- **Bloque Decoder Attention**

Q = el vector de la palabra procesándose en ese instante de la secuencia de salida

K = todos los vectores de palabras con información posicional de la secuencia de salida

V = todos los vectores de palabras con información posicional de la secuencia de salida

- **Bloque Encoder-Decoder Attention**

Q = el vector de salida de la máscara de atención del descodificador

K = todos los vectores del estado oculto del codificador

V = todos los vectores del estado oculto del codificador

Como se puede advertir en el listado anterior de características de cada bloque de atención, los *valores* son iguales que las *claves* $K = V$. Esto parece contradecirse con la intuición de la Sección 2.7 donde se afirmó que la atención se utiliza para relacionar una secuencia de entrada con otra de salida (por ejemplo, una frase en un idioma y en otro). Sin embargo, el mecanismo de *self-attention* se utiliza para relacionar las diferentes posiciones de la misma secuencia de entrada entre sí (de

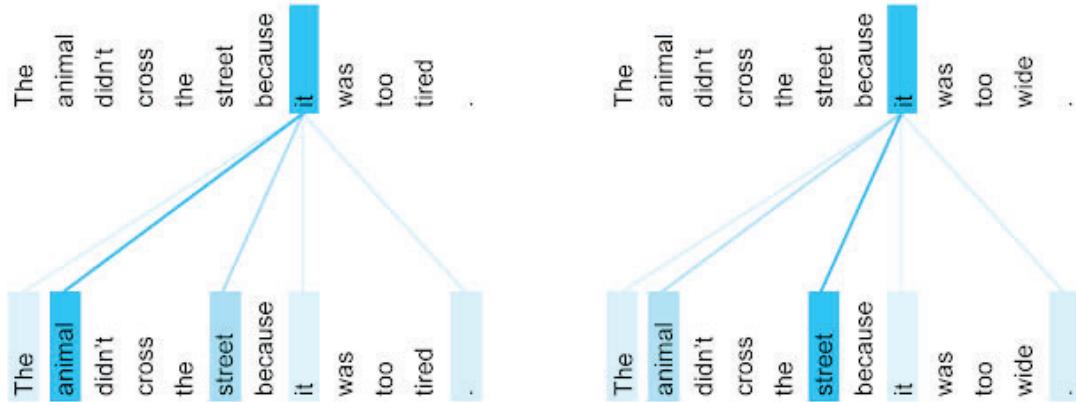


Figura 2.14: Pesos de atención que relacionan el pronombre *it* con *animal* y *street*, capturando la posible ambigüedad semántica (Zhang et al., 2020a)

ahí el nombre). De este modo, se sustituye la secuencia objetivo (valores) por la misma secuencia de entrada (claves), dando lugar a las relaciones entre palabras de una misma frase como las que pueden observarse en la Figura 2.14.

Llegado este punto de la explicación, se puede ya comprender que, dado que se toman los valores en cada estado de la red directamente (ponderadas dinámicamente usando la Atención), ya no se necesita propagar y combinar las señales de la secuencia hasta el estado oculto final, como ocurría en las Redes de Neuronas Recursivas. Por ello, se prescinde de ellas.

Sin embargo, los vectores de atención por sí solos son esencialmente un conjunto de multiplicaciones de matrices y, por tanto, transformaciones lineales. Para que la red aprenda características jerárquicas, es decir, para que aproxime funciones complejas, se requiere neuronas con funciones de activación no lineal.

Para ello, tras la capa de atención, se introducen transformaciones no lineales mediante una red de neuronas alimentada hacia delante con función de activación ReLu (Nair and Hinton, 2010). Los vectores de salida de la Red de Neuronas, de alguna manera, sustituyen los estados ocultos del codificador recursivo.

Sin embargo, esta arquitectura no conserva la influencia del orden de los símbolos en el significado. Por ejemplo, el estado oculto final para la frase "Los perros persiguen a los gatos" sería el mismo a "Los gatos persiguen a los perros". Por lo tanto, se ha de hallar un método que no ignore el orden de la secuencia para que el descodificador diferencie entre dos secuencias con entradas idénticas, pero con un orden diferente. Para ello, se introduce la codificación posicional.

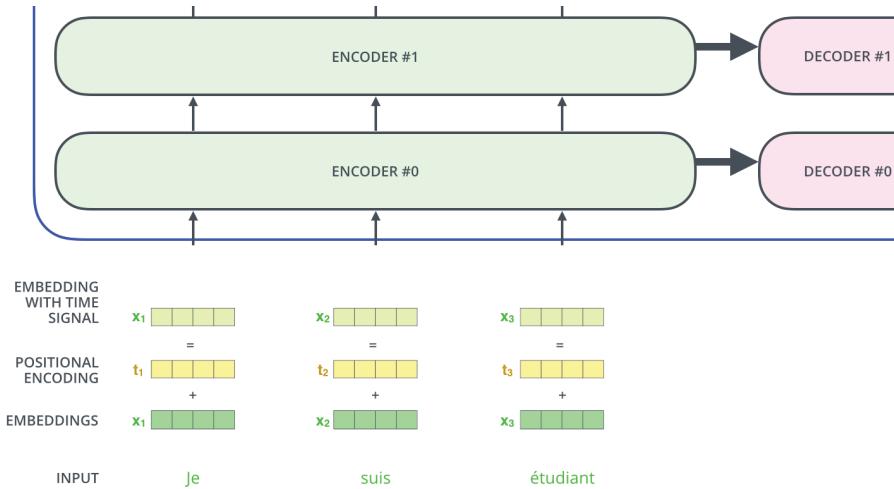


Figura 2.15: Codificación posicional (Alammar, 2019)

2.8.3. Codificación posicional

Como se ha dicho, el Transformer prescinde de recursividad. Para introducir la información de la posición de una palabra, se utiliza una codificación posicional.

Para conseguir esta codificación, en primer lugar, se obtiene el vocabulario del conjunto de entrenamiento. A cada palabra se le asigna un entero. Así, dada una secuencia de entrada se convierte cada palabra en uno de los valores enteros. A estas sencillas *word embeddings* se añade una información posicional que viene dada por la expresión:

$$\begin{aligned} PE_{(pos,2i)} &= \sin \left(pos / 10000^{2i/d_{\text{model}}} \right) \\ PE_{(pos,2i+1)} &= \cos \left(pos / 10000^{2i/d_{\text{model}}} \right) \end{aligned} \quad (2.17)$$

La intuición bajo esta codificación es que aportar distancias en las palabras permite diferenciar significados una vez que se proyectan en los vectores $Q/K/V$ mediante la multiplicación del mecanismo de atención.

Este proceso de codificación se observa en la Figura 2.15.

2.8.4. MultiHead Attention

La arquitectura mejora el funcionamiento de *self-attention* añadiendo un mecanismo denominado *multi-headed attention*. Básicamente, consiste en añadir varias unidades ("cabezas") de *self-attention*. Esta idea mejora el rendimiento de la capa de atención de dos maneras:

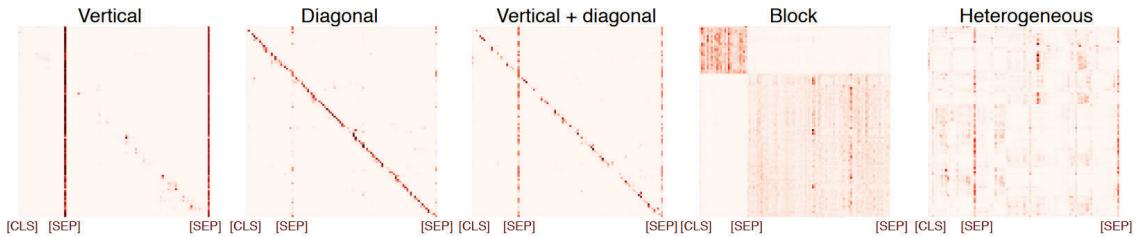


Figura 2.16: Tipos de *self-attention* utilizadas para entrenar una red neuronal. Los dos ejes de cada imagen representan los pesos de BERT, y los colores indican los pesos absolutos de atención (los colores más oscuros representan pesos mayores). De acuerdo con (Kovaleva et al., 2019), los tres primeros tipos están más probablemente asociados con el preentrenamiento, mientras que los dos últimos potencialmente codifican información semántica y sintáctica. (Kovaleva et al., 2019)

1. Aumenta la capacidad del modelo para centrarse en diferentes posiciones. Por ejemplo, si quiere representar la frase, "Un hombre y una mujer subieron la escalera; él casi se tropieza". Es posible que la palabra hombre tenga una fuerte relación con sí misma perdiendo la relación con "él". Añadiendo más capas se asegura que todas las posiciones se contemplen.
2. Permite a la capa de atención múltiples "subespacios de representación". Con varias capas de atención, se permiten múltiples conjuntos de matrices de pesos Consulta/Clave/Valor. El Transformer utiliza ocho "cabezas", por lo que hay ocho conjuntos de matrices para cada codificador/descodificador). Cada uno de estos conjuntos se inicializa de forma aleatoria. Luego, después del entrenamiento, cada conjunto se utiliza para proyectar los vectores de palabra de entrada en un subespacio de representación diferente. Cada uno de estos posibles espacios de representación dan lugar a atenciones especializadas distintas; se pueden observar en la Figura 2.16.

En la Figura 2.17b se observa un esquema del mecanismo *multihead attention*.

2.8.5. Descodificador

El codificador comienza procesando la secuencia de entrada. La salida del último codificador se transforma en un conjunto de vectores de atención K y V . Éstos se utilizan en la capa "Encoder-Decoder Attention" del descodificador, que le permiten centrarse en las posiciones más importantes de la secuencia de entrada.

Cada símbolo que genera el descodificador se introduce de nuevo en él; a este tipo de arquitectura se le denomina autorregresiva. Se repiten los cálculos hasta

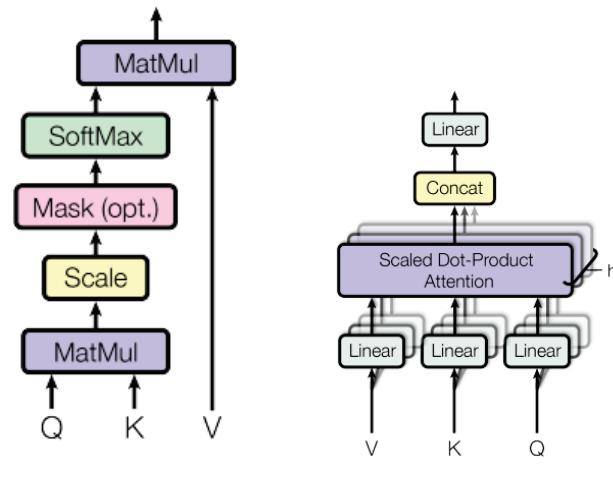


Figura 2.17: Atención en Transformers (Vaswani et al., 2017)

que se alcanza un símbolo especial que indica que el descodificador ha completado la generación de su salida. Los descodificadores inferiores del bloque "elevan" sus resultados de decodificación al igual que los codificadores. De modo semejante, se crean *word embeddings* y se añade codificación posicional en su entrada.

Las capas de *self-attention* en el descodificador funcionan de una manera ligeramente diferente a la del codificador:

En el descodificador, la capa de *self-attention* sólo puede atender a posiciones anteriores en la secuencia de salida. Esto lo permite una máscara (una matriz) en la que las posiciones futuras se establecen a $-\infty$ antes de pasarlas por la capa *softmax*.

2.8.6. Capa de salida

El bloque de descodificadores produce un vector de números en coma flotante. Para su decodificación en una palabra, se utiliza una última capa lineal, seguida por una capa de *softmax*.

La capa lineal es una red neuronal completamente conectada que proyecta el vector calculado por la pila de descodificadores en un vector más grande llamado *vector logits*.

Si el vocabulario del modelo son 10.000 palabras únicas ("vocabulario de salida"), el *vector logits* tendría 10.000 neuronas, cada una de las cuales corresponden con la puntuación de una palabra.

Esta puntuación se pasa por una capa *softmax* que convierte la convierte en probabilidades (todas positivas, suman 1.0) y se elige la neurona de mayor pro-

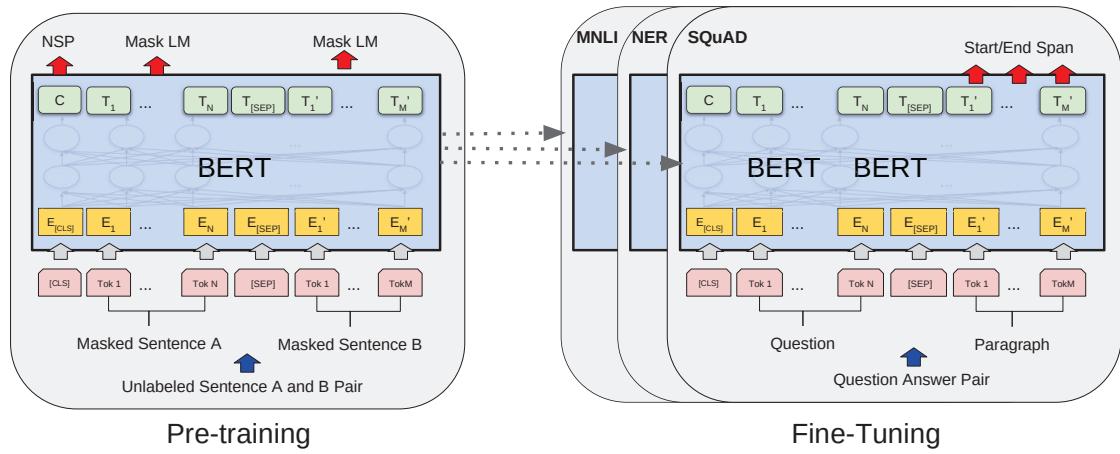


Figura 2.18: Las dos etapas del entrenamiento de BERT (Devlin et al., 2018)

babilidad, produciendo la palabra asociada a ella como resultado del cálculo en ese instante.

2.9. BERT

BERT (Pre-training of Deep Bidirectional Transformers for Language Understanding) es un modelo de representación del lenguaje de propósito general introducido por (Devlin et al., 2018). Los sistemas GPT-2 de OpenAI (basado en el Transformer ya explicado) (Radford et al., 2019) y ELMo (LSTM bidireccional) (Peters et al., 2018) habían demostrado que el rendimiento en varias tareas de Procesamiento de Lenguaje Natural mejora notablemente con el uso de modelos de lenguaje preentrenados. El gran beneficio de estos modelos radica en que su entrenamiento se ha realizado sobre datos no etiquetados, como los artículos de Wikipedia, grandes corpus de libros o noticias, lo que permite el entrenamiento sobre una enorme cantidad de datos. Gracias a ello, las redes aprenden una representación general del lenguaje natural que posteriormente se puede emplear para tareas específicas. A la primera etapa de aprendizaje de las propiedades lingüísticas a partir de datos no etiquetados se denomina "preentrenamiento". A la segunda de entrenar el modelo para una tarea específica (resumen, pregunta-respuesta, etc.) se denomina "ajuste fino".

Sin embargo, la unidireccionalidad de algunos modelos Transformer puede ser problemática en tareas como los sistemas pregunta-respuesta o resumen, donde es importante conocer el contexto de la frase en ambas direcciones. Por ejemplo, GPT-2 de OpenAI (Radford et al., 2019) utiliza un decodificador de Transformer, entrenado de izquierda a derecha, de modo que solo puede "mirar" las

palabras de su izquierda para generar el siguiente símbolo. Estos modelos se denominan autoregresivos y son aquellos en los que la salida del descodificador se utiliza como entrada del mismo (Sección 2.8.5). Por poner un ejemplo, en los contextos "fui al banco a depositar dinero en efectivo" y "fui al banco a sentarme", aunque el significado de la palabra "banco" es polisémica y sensible al contexto de su izquierda, GPT devolverá la misma representación para "banco". ELMo compensa este problema entrenando dos redes LSTM, una en cada dirección. Posteriormente, concatena las representaciones de las dos redes LSTM para obtener una representación semibidireccional. Sin embargo, tiene los problemas de las redes recursivas que se han tratado al comienzo de la Sección 2.8.

Así, frente a los modelos autoregresivos surge BERT, que es un *Denoising Autoencoder*: el objetivo del modelo es "reparar" una entrada a la que se ha introducido ruido.

Por lo tanto, los inconvenientes de la unidireccionalidad quedan solventados mediante un entrenamiento de eliminación de ruido que los autores han denominado "modelo lingüístico con máscaras", inspirado en el problema de "rellenar los huecos". El modelo de lenguaje sustituye aleatoriamente algunas palabras por una máscara, y el objetivo del modelo es predecir el identificador original del símbolo que ha sido enmascarado, teniendo en cuenta el contexto en que aparece. Ya no se reconstruye la secuencia de salida entera, sino que se predice solo los símbolos que han sido enmascarados.

Formalmente, las representaciones unidireccionales, al igual que las *word embeddings* de Word2Vec vistas en las Secciones 2.2 y 2.2, son una función $f(x)$ donde x es la pieza del vocabulario que da como resultado siempre la misma salida: cada palabra queda representada únicamente. Sin embargo, dada la polisemia del lenguaje natural es necesario disponer de distintas representaciones de la misma pieza en función del contexto en el que se encuentre. Esto es fundamental para poder hacer resumen abstractivo y es el objetivo de BERT.

2.9.1. Representación de la entrada

Para hacer que BERT pueda abordar variedad de tareas en la etapa de ajuste fino, al mismo tiempo que representa las palabras en su contexto bidireccional, el vector de entrada debe ser capaz de representar tanto una sola frase como un par de frases (por ejemplo, una pregunta y su respuesta). Una "secuencia" se refiere a la serie ordenada de símbolos de entrada, que puede ser una sola oración o dos oraciones juntas.

El primer símbolo de toda secuencia es siempre una pieza especial para clasificación [CLS]. El estado final correspondiente con este símbolo se utiliza como

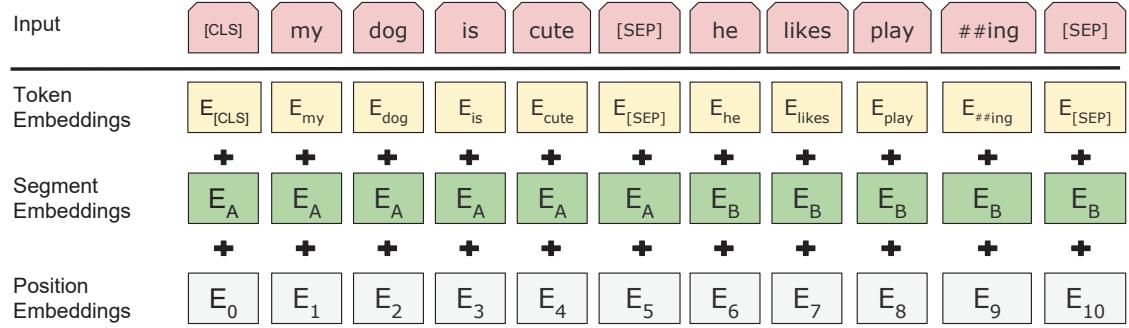


Figura 2.19: Construcción de los vectores de entrada de BERT (Devlin et al., 2018)

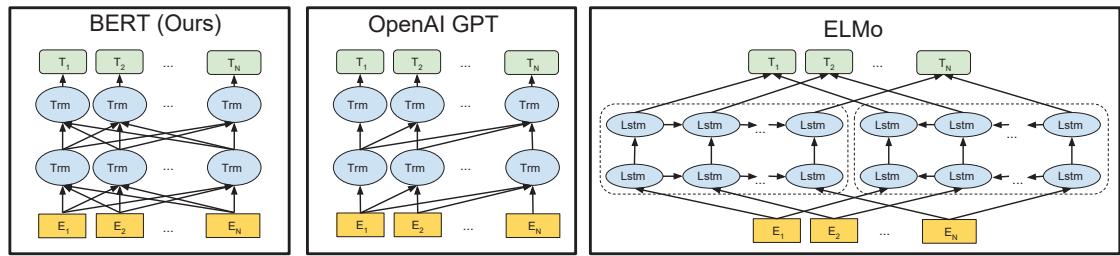


Figura 2.20: Comparativa de BERT con otros modelos basados en Transformers (Devlin et al., 2018)

la representación agregada de la secuencia para tareas de clasificación. Los pares de frases de entrada se juntan en una sola. La separación entre las frases se establece mediante el símbolo [SEP] y mediante la adición de un vector aprendido asignado a cada símbolo que indica si pertenece a la frase A o a la frase B. Como se puede observar en la Figura 2.19, se denota el vector de entrada como E , el vector del estado oculto final del símbolo [CLS] como $C \in \mathbb{R}^H$ y el vector final del i -ésimo símbolo de entrada como $T_i \in \mathbb{R}^H$.

Para un símbolo individual, su representación de entrada se construye sumando los vectores del segmento, la posición y el identificador como puede visualizarse en la Figura 2.19.

2.9.2. Arquitectura

BERT es idéntico al bloque de codificadores de Transformer (Vaswani et al., 2017) (Sección 2.8, Figura 2.13). Sin embargo, la diferencia fundamental es que el Transformer de BERT utiliza *self-attention* bidireccional, mientras que el Transformer GPT utiliza una *self-attention* en la que cada símbolo sólo puede atender al contexto a su izquierda.

2.9.3. Preentrenamiento

BERT ha sido preentrenado utilizando dos tareas no supervisadas.

Tarea 1. Modelo de lenguaje mediante máscaras Intuitivamente, es razonable creer que un modelo bidireccional profundo es más rico que uno que solamente sea de izquierda-derecha o la concatenación superficial de un modelo de izquierda-derecha y de derecha-izquierda. Como se ha dicho en la Sección 2.5, los modelos estándar de generación de lenguaje condicionada sólo pueden ser entrenados de izquierda a derecha o de derecha a izquierda, ya que el condicionamiento bidireccional permitiría que cada palabra se “viera a sí misma” directamente, y el modelo podría predecir trivialmente la palabra objetivo.

Para entrenar una representación bidireccional profunda, BERT enmascara algún porcentaje de los símbolos de entrada al azar, y luego el modelo se encarga de predecir esos símbolos enmascarados. Este procedimiento se denomina como como ‘Modelado de Lenguaje Enmascarado’ (MLM): los vectores ocultos finales correspondientes a los símbolos enmascarados se introducen en una capa *softmax* de salida sobre el vocabulario, como en un modelo de lenguaje estándar. Los autores de BERT, enmascaran el 15 % de todas los símbolos. Sólo se predicen las palabras enmascaradas en lugar de reconstruir toda la entrada.

Tarea 2. Predicción de la siguiente frase Los sistemas pregunta-respuesta o la inferencia de lenguaje, se basan en comprender la relación entre dos frases. Con el fin de entrenar un modelo que entienda las relaciones de las oraciones, se preentrena para una tarea de predicción de la siguiente frase. Esta tarea puede diseñarse trivialmente a partir de cualquier corpus monolingüe: dadas dos frases A y B, se sabe que A sigue a B o no. Para el entrenamiento, se preparan las entradas para que el 50 % de las veces no sean la siguiente.

2.9.4. Ajuste fino

Una vez se ha preentrenado BERT, se dispone de una red que puede representar secuencias lingüísticas y pares de secuencias. Por lo tanto, utilizar BERT para una tarea específica es tan sencillo como entrenar, sobre el modelo preentrenado, con los datos del problema en cuestión. Por ejemplo, para los sistemas pregunta-respuesta se introduce la pregunta, el separador [SEP] y la respuesta. Para clasificación de textos, se entrena sobre los textos del campo específico que se quieren clasificar, etc.

Sin embargo, para un problema como el resumen la aproximación de ajuste

fino no es trivial. Se tiene un codificador de Transformer, BERT, el cual es capaz de representar el lenguaje bidireccionalmente, propiedad que es muy adecuada para el resumen. Sin embargo, el resumen (como la traducción) es una tarea de texto a texto: se necesita además de un modelo de lenguaje, un decodificador. El diseño de la arquitectura se tratará en la Capítulo 4.

Inconvenientes de BERT para resumen

BERT presenta dos grandes inconvenientes. El primero es la gran cantidad de recursos que necesita su entrenamiento, que es común a todos los Transformer. El segundo es que no acepta secuencias de texto superiores a la dimensión para la que ha sido entrenado, 512 símbolos.

Teóricamente no hay nada que restrinja a un modelo Transformer a tener una longitud mayor de la secuencia de entrada. En la práctica, hay limitaciones de recursos, especialmente en la complejidad de la memoria, que es cuadrática en términos de longitud de la secuencia al aplicar un mecanismo de *self-attention*.

Se han investigado distintas soluciones para superar esta limitación. Yang et al. (2019) propone, para un sistema de pregunta-respuesta basada en artículos de Wikipedia, una aproximación jerárquica en la que se divide el texto en párrafos o frases, y luego se entrena utilizando estas unidades pequeñas; sin embargo, esta técnica no se puede aplicar a resumen, pues la respuesta a una pregunta puede encontrarse en esas unidades, pero la semántica de un texto exige la compresión unificada de todos los párrafos.

Por otro lado, dado que los Transformer tienen la posibilidad teórica de aprender las dependencias a largo plazo, Dai et al. (2019) proponen una arquitectura Transformer-XL que permite el aprendizaje de las dependencias más allá de una longitud fija sin alterar la coherencia temporal. Consiste en un mecanismo de recurrencia a nivel del segmento combinado con un esquema de codificación posicional. Es decir, añade recurrencia al modelo Transformer. De acuerdo con los autores, el método no sólo permite capturar la dependencia a largo plazo, sino que también resuelve el problema de fragmentación del contexto. Sin embargo, Transformer-XL está ideado en el contexto del modelado del lenguaje (predicir el siguiente símbolo) y no en un modelo de texto a texto, de manera que no se puede aplicar al problema de resumen. Además, el objetivo de este trabajo es la utilización de BERT, ya que es el único modelo disponible hasta el momento en Español, de modo que habría que hallar un modo de utilizar BERT en secuencias de gran longitud.

En esta línea se encuentra el trabajo de Zhang et al. (2020b), quienes utilizan BERT en el codificador y entranan un decodificador Transformer desde cero.

Para la segmentación de la entrada, las dos opciones inmediatas son:

1. Truncar la entrada al tamaño máximo de secuencia. Esto empeoraría el rendimiento ya que el modelo será incapaz de capturar las dependencias a largo plazo y la información global repartida en el texto.
2. Extender el tamaño de entrada del Transformer. Esto costaría una enorme cantidad de tiempo y recursos computacionales. Aunque teóricamente no hay nada que restrinja a un Transformer tener una mayor longitud de secuencia, en la práctica hay limitaciones de recursos, especialmente la complejidad de la memoria al calcular la *self-attention*, que es cuadrática en términos de longitud de la secuencia.
3. Hallar algún medio de segmentación que permita capturar las dependencias a largo plazo de 512 en 512 símbolos (Pappagari et al., 2019). La solución para este problema se tratará en la Sección 4.2.1.

2.10. Sistema de evaluación de resumen

La evaluación de los resúmenes es fundamental por varias razones (por ejemplo, hacer un seguimiento del progreso de entrenamiento o comparar los resultados con los de otras investigaciones). La evaluación humana, aunque es inmediata y obvia, no termina de ser efectiva porque es lenta, cara y los resultados de distintos evaluadores pueden no ser comparables. Para evaluar automáticamente los resúmenes, se ha convertido en un estándar la métrica *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) (Lin, 2004), que trata de correlacionar un resumen de referencia y el generado automáticamente. Se utilizarán 3 métricas.

1. ROUGE-1: que calcula el solapamiento de palabra a palabra (unígrafo) entre el resumen candidato y el resumen de referencia.
2. ROUGE-2: es similar a ROUGE-1. Compara pares de palabras entre el resumen candidato y resumen de referencia (bigramas).
3. ROUGE-L: utiliza la subsecuencia común más larga para calcular la puntuación. Una subsecuencia no debe ser confundida con una subcadena. Para las frases "Mi casa está en Madrid" y "Mi casa está en Rivas Madrid", la subsecuencia común más larga sería "Mi casa está en Madrid", mientras que la subcadena común más larga sería sólo "Mi casa está en".

Cuando se compara el solapamiento de dos cadenas se ha de diferenciar entre precisión y exhaustividad. La exhaustividad indica cuánta información del resumen de referencia está presente en el resumen del candidato, la precisión indica cuánta de la información del resumen candidato está realmente presente en el resumen de referencia.

Es común combinar estas dos medidas en una sola, llamada F1. Se calcula mediante la siguiente fórmula:

$$F_1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \quad (2.18)$$

Capítulo 3

Estudio y descripción del conjunto de datos

En este capítulo se presenta el conjunto de datos de libros etiquetados. Se estudia y prepara el conjunto para tareas de clasificación, desambiguación de autor, generación de títulos y, sobre todo, resumen abstractivo.

El conjunto de datos consta de 44552 libros con sus metadatos. Están descritos en 11 columnas que corresponden con el número (identificador), el título, autor, género, colección, sinopsis, año de publicación, páginas, idioma, valoración y números de votos dados por los usuarios de la plataforma del proyecto Scriptorium. Se han obtenido de la web epublibre.org por medio de técnicas de *scrapping*. Cada fila es un libro etiquetado con estos datos. En las Tablas 3.1, 3.2 y 3.3 se observa la forma del conjunto.

En la Tabla 3.4 se puede ver la descripción de los datos numéricos. Estos son el *año*, el *número de páginas* y la *valoración* de cada libro. En lo que respecta al *año*, el libro más antiguo pertenece al 2500 antes de nuestra era; el más reciente al 2019. La media del año de publicación es 1962. En cuanto al *número de páginas*, se observa que el libro con mayor longitud tiene 12665, mientras que la media está en 256 páginas. Lo más interesante de esta Tabla son quizás las *valoraciones*. El libro

Tabla 3.1: Forma del conjunto de datos: Título, Autor, Géneros, Colección

Número	Título	Autor	Géneros	Colección
26947	La plaza del Diamante	Mercè Rodoreda	Drama	NaN
27603	Dubrovsky	Aleksandr S. Pushkin	Otros	NaN
43008	Di adiós al mañana	Horace McCoy	Policial	NaN
43875	Los inmigrantes	Howard Fast	Histórico	La familia
30251	Sin conciencia	Robert D. Hare	Psicología	NaN
43582	Limericks (El libro del Sinsentido)	Edward Lear	Humor	NaN

Tabla 3.2: Forma del conjunto de datos: Sinopsis

Número	Sinopsis
26947	La plaza del Diamante (título original en catalán)
27603	Obra incompleta de Pushkin escrita en 1832 y publicada en 1841
43008	Di adiós al mañana es una obra maestra. McCoy dice que es la mejor novela de la literatura universal
43875	Howard Fast nos entrega, en su obra Los inmigrantes, una visión realista y cruda de la vida en Estados Unidos
30251	A la mayoría de la gente le atraen y a la vez despiertan repulsión
43582	Edward Lear (1812 - 1888), autor inglés. Genial poeta y pintor

Tabla 3.3: Forma del conjunto de datos: Año, Páginas, Valoración, Nº votos

Número	Año publicación	Páginas	Idioma	Valoración	Nº Votos
26947	1962	182	Español	8.6	20
27603	1841	94	Español	9.7	7
43008	1948	277	Español	10.0	1
43875	1977	343	Español	8.9	15
30251	1993	227	Español	9.8	8
43582	1846	240	Español	7.0	2

más votado tiene 1246 votos; sin embargo, la media está en 14 votos.

En la Tabla 3.5 se puede ver la descripción de los datos textuales del conjunto. Son *título*, *autor* e *idioma*. De momento, no se trata otro tipo de dato textual, el *género*, porque será abordado individualmente dada su importancia para tareas de clasificación. Hay 44552 títulos, de los cuales 42629 son únicos. En cuanto a los autores, cabe reparar que, aunque hay 44552 títulos, estos están escritos por 13656 escritores. Se repite hasta 50 veces el título de "Cuentos completos". El mayor interés de esta tabla radica en los idiomas. El conjunto de datos tiene libros en 11 idiomas, siendo el español el dominante con 40037 libros.

Tabla 3.4: Descripción de los datos numéricos del conjunto

	Año publicación	Páginas	Valoración	Nº Votos
count	44.552	44.552	43.229	44.552
mean	1.962	256	8	14
std	199	208	1	34
min	-2.500	1	1	0
25 %	1.967	135	8	4
50 %	1.997	221	9	8
75 %	2.011	326	9	14
max	2.019	12.665	10	1.246

Tabla 3.5: Descripción de los datos de texto del conjunto

	Título	Autor	Idioma
count	44552	44552	44552
unique	42629	13656	11
top	Cuentos completos	Corín Tellado	Español
freq	50	392	40037

Tabla 3.6: Descripción de los datos del campo Idioma

Idioma	Nº libros	Porcentaje
Español	40037	89,87 %
Catalán	2696	6,05 %
Inglés	1520	3,41 %
Francés	90	0,20 %
Alemán	79	0,18 %
Gallego	51	0,11 %
Euskera	26	0,06 %
Italiano	21	0,05 %
Esperanto	12	0,03 %
Portugués	10	0,02 %
Sueco	10	0,02 %

En la Tabla 3.6 se muestra número y el porcentaje de libros por idioma. El español abarca mayoritariamente un 89,87 % de los libros, seguido del catalán con 6,05 % y el inglés con 3,41 %.

El campo *género* requiere un estudio especial dada su importancia en las tareas de clasificación de textos. Los libros se ubican en 42 géneros cuya distribución puede contemplarse en la Tabla 3.7 y en la Figura 3.1. La mayoría son libros de género de detectives (5371), policial (4640), histórico (3859), drama (3726) y fantástico (3680), los cuales suponen el 47 % del conjunto, dejando el restante 53 % a los demás 37 géneros. Los géneros menos frecuentes son *idiomas*, *padres*, *encyclopedia* (entorno a las dos o cinco decenas); quizás se puedan despreciar para determinados problemas, dada su diferencia de magnitud frente a los géneros más frecuentes (orden de miles).

En el Anexo A pueden observarse las palabras más frecuentes por cada género. Es reseñable que cada uno de ellos presenta palabras exclusivas: por ejemplo, *cine* o *música* son palabras muy frecuentes en la categoría *arte* y casi inexistentes en las demás; lo mismo ocurre en con *física* y *matemáticas* en *exactas*. Por ello, cabría formular la hipótesis de que una sencilla aproximación por TF-IDF daría

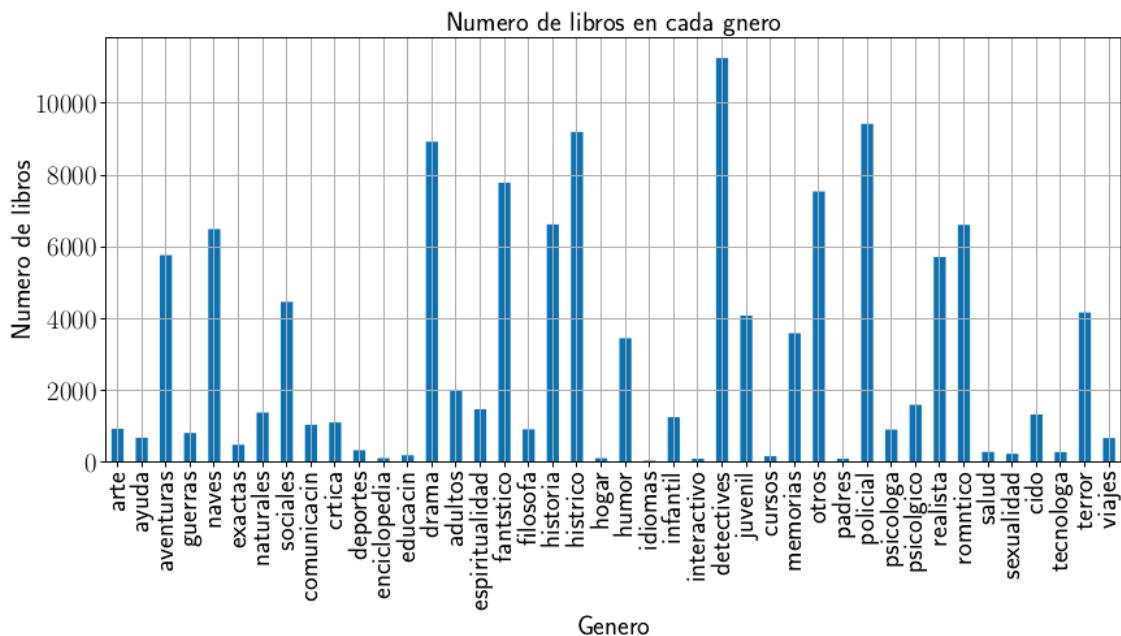


Figura 3.1: Número de libros por género

buenos resultados para la tarea de clasificación.

El conjunto de datos presenta la ventaja de que cada libro está etiquetado en más de un género. Esto permite su utilización en problemas más complejos como la clasificación multietiquetada. En la Figura 3.2 se contempla que 10563 libros tienen dos géneros asignados, 2408 tres, y 324 cuatro etiquetas.

Por último, se estudian los campos de *sinopsis* y contenido de los libros, sobre los que se centrará este trabajo. Son los campos más valiosos, ya que se tratan de resúmenes abstractivos realizados por humanos (los editores de los libros) alineados con su texto completo.

En la Tabla 3.8 se observa que la sinopsis con más palabras tiene 2603, la que menos 8 y la media se encuentra en 154. En cuanto al contenido del libro, el más largo tiene 3799500 palabras y el más corto, 300. En las Figuras 3.3 y 3.4 pueden observarse las distribuciones de la longitud de la sinopsis y los libros.

Tabla 3.7: Número de libros por género

	Géneros	Nº libros
1	detectives	5371
2	policial	4640
3	histórico	3859
4	drama	3726
5	fantástico	3680
6	otros	3237
7	naves	3174
8	romántico	3123
9	aventuras	2785
10	historia	2672
11	realista	2325
12	juvenil	2229
13	terror	1993
14	sociales	1700
15	humor	1467
16	memorias	1412
17	filosofía	1235
18	adultos	892
19	infantil	791
20	psicológico	665
21	espiritualidad	579
22	naturales	561
23	ácido	541
24	crítica	472
25	arte	418
26	comunicación	417
27	psicología	365
28	guerras	365
29	ayuda	301
30	viajes	267
31	exactas	194
32	deportes	131
33	salud	117
34	tecnología	112
35	sexualidad	97
36	educación	91
37	cursos	79
38	hogar	53
39	encyclopedia	53
40	interactivo	52
41	padres	43
42	idiomas	20

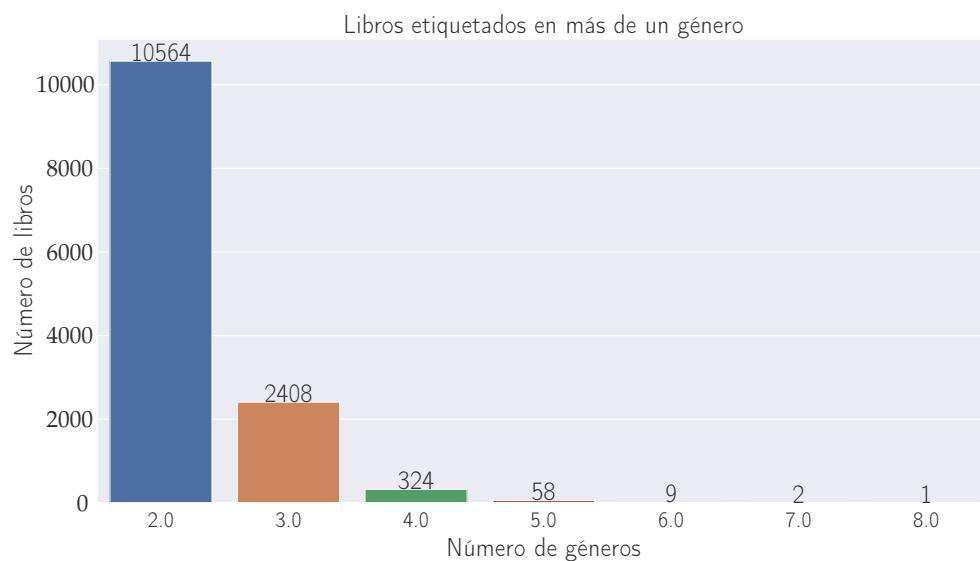


Figura 3.2: Libros con más de un género asignado

Tabla 3.8: Descripción de los campos del número de símbolos de cada libro y los resúmenes (unidades: símbolos)

	Libro	Sinopsis
count	44552	44552
mean	76772	166
std	62379	85
min	300	8
25 %	40500	111
50 %	66300	154
75 %	97800	206
max	3799500	2603

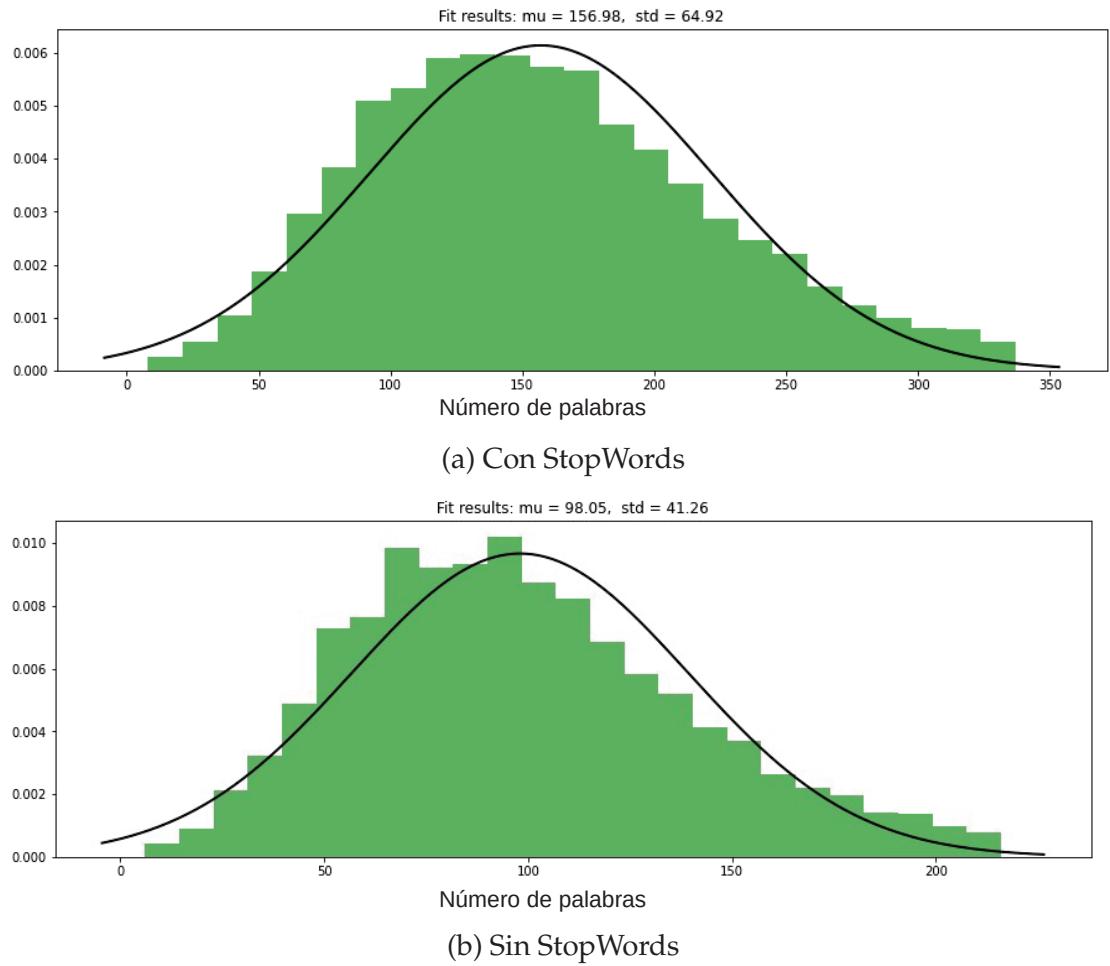


Figura 3.3: Distribución de la longitud de las sinopsis

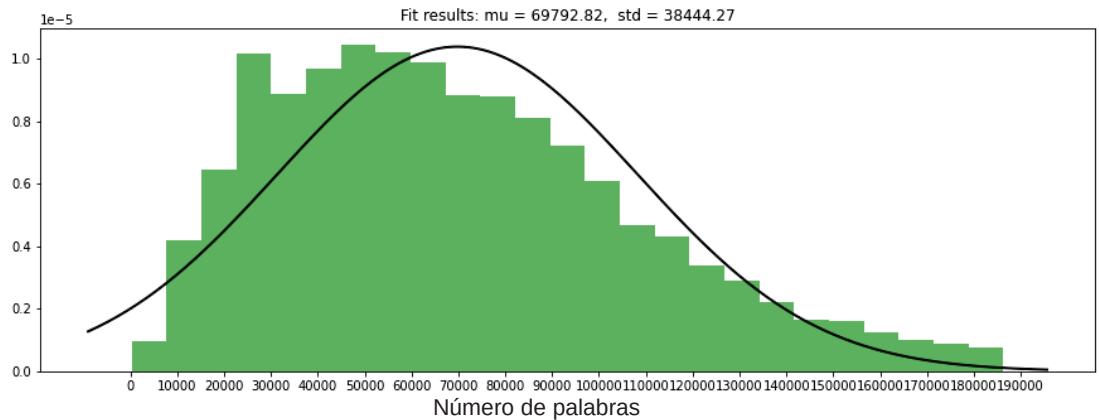


Figura 3.4: Distribución de la longitud de los libros

Capítulo 4

Modelo propuesto

Partiendo del conocimiento expuesto en el Capítulo 2, se van a detallar la arquitectura y la preparación de datos utilizadas.

4.1. Arquitectura

Se implementa una arquitectura codificador-descodificador como la vista en la Sección 2.5. Para el bloque codificador se utiliza BERT. Para descodificador, el bloque Transformer normal. La arquitectura es semejante a la propuesta por Zhang et al. (2019a).

Como se ha expuesto en la Sección 2.9, BERT permite una o dos frases de entrada, en función de la tarea de ajuste fino para la que va a ser utilizado. Para este trabajo, solamente se requiere una frase: el texto para resumir. El descodificador Transformer utiliza mecanismos de atención empleando las representaciones de salida del codificador. La arquitectura completa puede verse en la Figura 4.1.

4.1.1. Elección del codificador

La elección del codificador preentrenado es fundamental para obtener buen rendimiento. En la Tabla 4.1 se contemplan los distintos modelos BERT preentrenados disponibles para su descarga. De acuerdo con Devlin et al. (2018), el modelo publicado de mayor tamaño, $BERT_{large}$, ofrece en general mejores resultados, pero solo está disponible en inglés y requiere una gran cantidad de memoria y tiempo de GPU. Otra posibilidad hubiera sido la utilización del modelo $BERT_{multilingual}$; sin embargo, se compone de más de 100000 piezas de vocabulario de 100 idiomas. Dado que el corpus de entrenamiento está en español, sería un malgasto de recursos utilizar un vocabulario de ese tamaño cuando solo se empleará una pequeña parte. Sin embargo, Google no ofrece ningún modelo en-

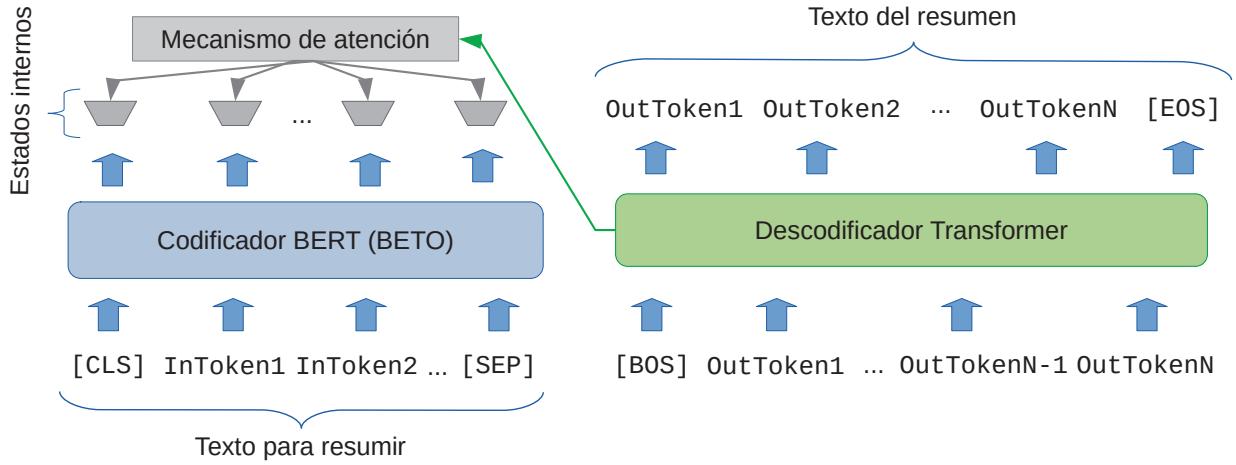


Figura 4.1: Arquitectura propuesta

Tabla 4.1: Descripción de los distintos modelos de BERT (Devlin et al., 2018; Cañete et al., 2020)

Model	Details of the model
bert-base-uncased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.
bert-large-uncased	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on lower-cased English text.
bert-base-cased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased English text.
bert-large-cased	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on cased English text.
bert-base-multilingual-uncased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased text in the top 102 languages with the largest Wikipedias
bert-base-multilingual-cased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased text in the top 104 languages with the largest Wikipedias
beto-base-uncased	12-layer, 1024-hidden, 16-heads, 110M parameters. Trained on lower-cased Spanish text.
beto-base-cased	12-layer, 1024-hidden, 16-heads, 110M parameters. Trained on lower-cased Spanish text.

trenado exclusivamente en español. Por lo tanto, se eligió BETO (Cañete et al., 2020): es un modelo *BERT_{base}* no oficial que ha sido entrenado sobre el texto de Wikipedia y de las noticias de distintos medios en español. El vocabulario utilizado es el de *BETO – uncased* de 31002 palabras.

4.2. Preparación de los datos para resumen

La preparación de los datos consta de dos tareas. La primera consiste en segmentar el texto para adaptar la entrada y salida al número de símbolos permitidos por la arquitectura. La segunda incluye las subtareas habituales de tratamiento de texto en lenguaje natural: normalización y separación de los símbolos de la secuencia de entrada, en particular por medio de WordPiece (Kudo and Richardson, 2018). Merece la pena explicar brevemente este tipo especial de separación de secuencias de símbolos (*tokenización*): el codificador BERT ha sido entrenado sobre un conjunto de datos y un vocabulario fijo (31002 símbolos). Con WordPiece, se pueden separar los símbolos de la entrada de tal manera que se cubra el espectro de palabras que se encuentran fuera del vocabulario de preentrenamiento, *Out-Of-Vocabulary (OOV) words*, en la terminología habitual en inglés. Por ejemplo, ante la palabra fuera del vocabulario *playing*, se separa como dos unidades distintas: *play* y *#ing*.

El propio separador de símbolos de BERT se encarga de realizar esta segunda tarea. A continuación, se ahondará en la segmentación, asunto también clave para el correcto desempeño de la red.

4.2.1. Segmentación de la entrada

Para la preparación del texto de entrada, hay dos enfoques posibles: o bien se adapta el tamaño de la red al conjunto de entrenamiento, o bien se adapta el conjunto de entrenamiento a la red.

Para el primer enfoque, la recomendación es tomar la secuencia más larga del conjunto de entrenamiento y establecer ese tamaño como el número de neuronas de entrada. Para el problema que nos ocupa, es inviable por dos razones:

1. Calcular la secuencia de símbolos más larga sobre un conjunto de más de 5 gigabytes de texto requiere una gran cantidad de tiempo, y, además, el resultado sería previsiblemente intratable. Determinados libros, como el caso de *San Camilo 1936* de Camilo José Cela o *Solar bones* de Mike McComarck, se componen de una sola frase a lo largo de 200 páginas. La complejidad de

la memoria al aplicar *self-attention* es cuadrática en términos de longitud de la secuencia.

2. La razón indiscutible es que BERT ya ha sido preentreado para un tamaño de 512 símbolos de entrada. Viene ya dado por las exigencias de partida del modelo. Sí que permite, sin embargo, y como es de esperar, entradas menores.

Por lo tanto, el único enfoque posible es el segundo: adaptar el conjunto al modelo.

Como puede verse en la Tabla 3.8, la mayoría de las sinopsis tienen un tamaño que se encuentra entorno a los 200 símbolos. Como la entrada se va a alinear con la salida, es una decisión de diseño razonable el segmentar la entrada de 256 en 256 símbolos.

La aproximación más sencilla es truncar la secuencia, es decir, dado un libro de varios miles de símbolos, tomar solamente los 200 primeros; sin embargo, de manera inmediata se concluye que así se perderían todas las dependencias de significado que hacen posible un resumen. Por lo tanto, surge la necesidad de utilizar todo el texto, o la mayor parte del este, para que la red capture el significado del libro y pueda resumirlo.

(Pappagari et al., 2019) propone la siguiente segmentación del texto: se toma una ventana de un número determinado de símbolos, en nuestro caso 200, y se utiliza para trocear el texto manteniendo una cantidad de símbolos en la siguiente secuencia. Por ejemplo, si la ventana consta de 200 símbolos se desplaza de 150 en 150. Así, se consigue que, en entradas alimentadas sucesivamente, se conserven algunas relaciones textuales. Este método se puede contemplar más claramente en la Figura 4.2.

4.3. Preparación de datos para generación de sinopsis

La tarea de generar un resumen a partir del texto entero de un libro es muy ambiciosa. Para probar el rendimiento en secuencias cuyas dependencias a largo plazo se puedan tratar en una sola entrada, se propone otro experimento: dado un título, se genera su sinopsis. Con este fin, es estrictamente necesario tomar solamente las sinopsis de un tamaño inferior a 256 símbolos.

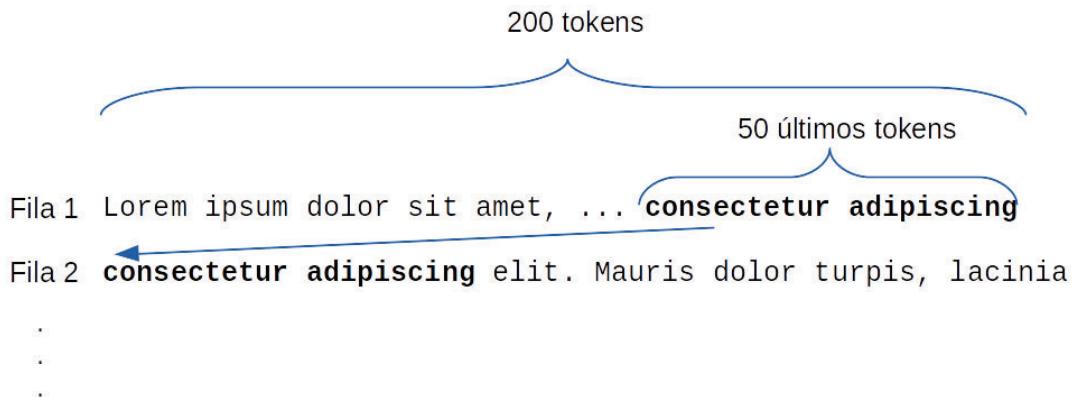


Figura 4.2: Segmentación del texto

4.4. Implementación

Se ha utilizado la implementación disponible GitHub¹; esta emplea Python como lenguaje de programación y se apoya en la librería de aprendizaje automático sobre Transformers de Huggingface (Wolf et al., 2019).

Para este trabajo se ha modificado el código para hacer el entrenamiento de un conjunto de datos mayor.

En el capítulo que viene a continuación se detallaran los experimentos llevados a cabo con el modelo propuesto.

¹<https://github.com/kururuken/BERT-Transformer-for-Summarization>

Capítulo 5

Experimentos

Se han llevado a cabo dos experimentos. El primero, de resumen abstractivo. El segundo, de generación automática de sinopsis para un título dado. La motivación para la comparación de ambos experimentos radica en que, en el primero, el texto de entrada es un libro que supera el número de símbolos establecidos por el preentrenamiento de BERT, por lo que es necesario realizar algún tipo de segmentación, es decir, dividir un libro en varias entradas distintas; por el contrario, en el segundo experimento, ninguna de las entradas supera el límite de símbolos del modelo preentrenado: todas las dependencias se encuentran en los símbolos de una sola entrada. Ambos se han realizado bajo la misma configuración.

5.1. Configuración

En todos los experimentos se ha utilizado el modelo $BETO_{base}$ (Cañete et al., 2020), basado en $BERT_{base}$ (Sección 4.1.1) para el codificador. Este se compone de 12 capas ocultas y 12 *attention heads* utiliza un vector de representación de palabras de 768 y una red completamente conectada con 3072 neuronas en la capa oculta. El decodificador es prácticamente idéntico en los hiperparámetros. La gran diferencia radica en que no está preentrenado. El decodificador Transformer (Sección 2.8.5) también está compuesto de 12 capas ocultas y 12 *attention heads*.

Se ha entrenado utilizando el optimizador Adam con una tasa de aprendizaje de 3×10^{-4} , $\beta_1 = 0.9$ $\beta_2 = 0.999$ $\epsilon = 10^{-9}$. Se ha utilizado un ajuste dinámico de la tasa de aprendizaje (Vaswani et al., 2017):

$$learning_rate = d_{model}^{-0.5} \cdot \min \left(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5} \right) \quad (5.1)$$

la ecuación puede interpretarse como el incremento lineal de la tasa de aprendiza-

Tabla 5.1: Resumen de los parámetros utilizados

Parámetro	Valor
Función de activación	"gelu"
$P_{dropout}$	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
max_position_embeddings	256
num_attention_heads	12
num_hidden_layers	12
vocab_size	31002

je hasta que $warmup_steps < steps$, que es el momento en que va disminuyendo poco a poco. En los experimentos se ha establecido $warmup_steps = 4000$.

Para la regularización, se ha empleado *dropout* (Srivastava et al., 2014) con un factor de 0.1.

Durante el entrenamiento, se ha establecido el tamaño del lote de 40, el cual maximizaba la memoria disponible.

El tamaño máximo de la secuencia de entrada se ha establecido en 256, tal y como se ha discutido en la Sección 4.2.1.

Estos parámetros óptimos se han extraído de distintas pruebas experimentales y se pueden contemplar resumidos en la Tabla 5.1.

El ordenador donde se hicieron las pruebas estaba equipado con una tarjeta gráfica NVIDIA GeForce 2060 de 6GB; se observaron severas limitaciones de memoria, por lo que se requirió mayor capacidad de cómputo. Los datos que aquí se presentan son el resultado del entrenamiento del modelo en 4 GPUs Tesla.

5.2. Conjunto de datos

Para el primer experimento, dada la limitación del vector de entrada de BERT, se ha preparado el conjunto de datos de acuerdo con lo expuesto en (Pappagari et al., 2019) y desarrollado en la Sección 4.2. Se ha segmentado utilizando un tamaño de entrada de 256 tokens.

Tras la segmentación han resultado 3058933 ejemplos.

Se ha dividido el conjunto de datos en tres: entrenamiento (80 %), test (10 %) y evaluación (10 %). El conjunto de entrenamiento consta de 2505111 ejemplos, los de test y evaluación, 276911.

De forma análoga, para el segundo experimento, también se ha dividido el con-

junto de datos en tres: entrenamiento (80 %), test (10 %) y evaluación (10 %). El conjunto de entrenamiento consta de 31325 ejemplos, los de test y evaluación, 3915. Se ha llevado a cabo la preparación de la Sección 4.3, donde se han tomado solo los libros en español con una sinopsis por debajo de los 256 símbolos. Se analiza el corpus y se encuentra que 533 títulos tienen una sinopsis de tamaño superior. Dada la pequeña magnitud que representan, se descartan del conjunto.

5.3. Experimento 1 - Resumen abstractivo

Tras distintas pruebas, los resultados óptimos del modelo sobre el corpus de libros en español para la tarea de resumen abstractivo son los que se contempla en la Tabla 5.2. El modelo propuesto obtiene una puntuación ROUGE-1 de 49.43, ROUGE-2 de 12.86 y ROUGE-L de 41.76, cuya media es 34.35, sobre los datos de test. En la Figura 5.1 se contemplan la evolución de la puntuación ROUGE a lo largo del entrenamiento.

La interpretación de los resultados de la Tabla 5.2 es que el modelo aprende bien las palabras que son relevantes en el texto, es decir, hace un *keyword summarization* (Sección 2.1) aceptable (ROUGE-1 en torno a 49), pero falla al situarlas en el orden adecuado para generar un enunciado (ROUGE-2 bajo). Combinando ambos resultados se podría concluir que el sistema es bueno capturando la información relevante pero no generando texto suficientemente coherente.

Si estos datos se comparan con los resultados de la Tabla 5.3, que muestra las puntuaciones de los distintos modelos de aprendizaje profundo sobre el corpus CNN/Daily Mail, estándar para resumen abstractivo en inglés, se observan unas puntuaciones semejantes: ROUGE-1 es algo superior en el modelo de este trabajo mientras que ROUGE-2 es en torno a 5 puntos inferior. La media, no obstante, supera levemente al resto de modelos (alrededor de 31). Este tipo de resultados no se puede interpretar como una mejora, pues son conjuntos de datos de distinta naturaleza, pero sirve para obtener una base comparativa del desempeño habitual de estos modelos en resumen abstractivo, así como nos indican que la implementación y el entrenamiento han sido correctos y alcanzan valores próximos al estado del arte.

Por último, en la Tabla 5.4 se observan algunas producciones del resumen. El sistema devolvía un número mayor de frases que las expuestas. Para la Tabla, se han extraído manualmente las dos frases más significativas. Cabe reseñar que el resto eran ruido o repeticiones.

Tabla 5.2: Experimento 1 - Resultados de ROUGE F1 para el modelo propuesto sobre el conjunto de datos de test para resumen abstractivo

	ROUGE-1	ROUGE-2	ROUGE-L	AVG
Modelo BERT-Transformer propuesto	49.43	12.86	41.76	34.35

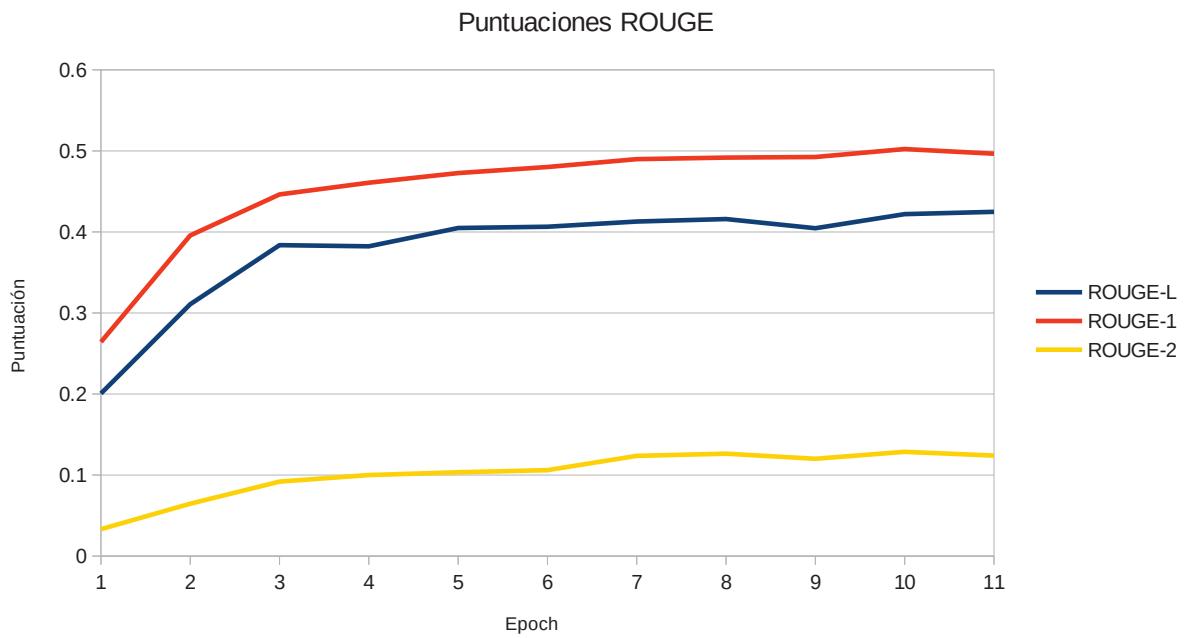


Figura 5.1: Experimento 1 - Resultados de ROUGE F1 para el modelo propuesto sobre el conjunto de datos de test a lo largo del entrenamiento

Tabla 5.3: Resultados de ROUGE F1 para varios modelos sobre el conjunto de datos de test CNN/Daily Mail. AVG es la media de ROUGE-1, ROUGE-2 y ROUGE-L. (Zhang et al., 2019a)

Abstractive	ROUGE-1	ROUGE-2	ROUGE-L	AVG
PointerGenerator+Coverage (See et al., 2017)	39.53	17.28	36.38	31.06
ML+RL+intra-attn (Paulus, 2019)	39.87	15.82	36.90	30.87
inconsistency loss (Hsu et al., 2018)	40.68	17.97	37.13	31.93
Bottom-Up Summarization (Gehrmann et al., 2018)	41.22	18.68	38.34	32.75
DCA (Celikyilmaz et al., 2018)	41.69	19.47	37.92	33.11
Modelo BERT-Transformer (Zhang et al., 2019a)	39.50	17.87	36.65	31.34

Tabla 5.4: Experimento 1 - Producciones de resúmenes para libros célebres

Libro	Resumen del modelo propuesto
Crimen y castigo	asesinado en su casa. a partir de ese momento, se vera envuelto en un torbellino de intrigas y venganzas.
El Señor de los Anillos	una familia que, como consecuencia de la maldicion de la tierra, el señor de los anillos. en el libro se narra la historia de el señor de los anillos, elfos y enanos, en el heroe de la lucha contra el malvado
Harry Potter	harry se ha convertido en el hombre mas peligroso y peligroso, y harry se ha convertido en un hombre. una serie de extraños sucesos que incluyen a harry potter y a harry potter, harry potter y a harry potter, harry potter
El Quijote	el clásico quijote, el quijote y el quijote, el quijote y el quijote, el quijote y el quijote. el quijote es el quijote quijote

5.4. Experimento 2 - Generación de sinopsis

Una vez se han planteado los problemas al procesar secuencias que superan el tamaño de la entrada de los modelos basados en BERT, se quiere probar el rendimiento para entradas cuyo número de símbolos no exija ser dividida.

Así, se diseña un nuevo experimento consistente en generar una sinopsis dado el título de una novela. A partir del corpus del Capítulo 3, se elabora un conjunto de datos Título-Sinopsis, donde no se superen los 256 símbolos de entrada.

En la Tabla 5.5, se contemplan las puntuaciones de ROUGE para este experimento. Cabe recordar que esta métrica indica cuánta información hay del resumen de referencia en el resumen generado por la inteligencia artificial. Para este problema, el modelo tiene que inferir todo el argumento a partir de las pocas palabras del título. Por ello, es normal que tenga peores resultados que el modelo para resumen. No obstante, lo más llamativo de este experimento es la evaluación cualitativa, pues estamos ante un modelo que está "creando" una sinopsis. A partir de un título cualquiera, inventa su argumento. En la Tabla 5.6 se contemplan algunas producciones. Cabe reseñar que los tres primeros títulos son inventados y, para los títulos existentes, crea una sinopsis apócrifa que guarda relación con

el significado de la secuencia de entrada. Dado el interés de este modelo, para divulgar sus curiosos resultados, se ha habilitado un bot en la red social Twitter: <https://twitter.com/apocrifosbot>. Los usuarios mencionan al bot con un título y este les responde con una sinopsis inventada o apócrifa.

Este experimento demuestra que BERT, para secuencias de entrada del tamaño del preentrenamiento funciona notablemente mejor. Prueba de ello es que, al leer las producciones de uno y otro modelo (Tabla 5.4 vs. Tabla 5.6), se observa que es capaz de adquirir las reglas del lenguaje con mayor competencia.

Tabla 5.5: Experimento 2 - Resultados de ROUGE F1 para el modelo propuesto sobre el conjunto de datos de test para generar sinopsis a partir del título

	ROUGE-1	ROUGE-2	ROUGE-L	AVG
Modelo BERT-Transformer propuesto	31.94	5.2	28.04	21.71

Tabla 5.6: Experimento 2 - Producciones del modelo para algunos títulos

Capítulo 6

Conclusiones y líneas futuras

Este trabajo ha presentado un nuevo conjunto de datos de textos literarios etiquetados. Tras su estudio y descripción, se han llegado a distintas conclusiones sobre su naturaleza: características de los datos numéricos y textuales, tales como la distribución de los géneros o la longitud media de las sinopsis.

El estudio de estas características, combinado con la investigación teórica del Capítulo 2, ha permitido tomar las decisiones de diseño del modelo. Así, se ha elegido una arquitectura codificador-descodificador basada en Transformers, cuyo codificador es BERT.

Se han llevado a cabo dos experimentos. El primero, para resumen abstractivo; el segundo para generación de sinopsis inventadas. Se han evaluado con la métrica ROUGE (Lin, 2005). Tras el estudio de los resultados del primer experimento, se observa que el sistema es bueno capturando la información relevante, es decir, efectuando un resumen de palabras clave, pero no generando texto suficientemente coherente. En cuanto al segundo experimento, se contempla que la red ha aprendido a generalizar la escritura de una sinopsis, llegando incluso a "crear" nuevos argumentos.

Los resultados para el Experimento 1 no han sido suficientemente satisfactorios debido a las limitaciones de los modelos de aprendizaje profundo para capturar las dependencias a largo plazo que se dan en grandes secuencias símbolos (Hochreiter et al., 2001) y, también, al pequeño tamaño del modelo. Sin embargo, aunque los resultados parecen coincidir con Tran et al. (2018), quienes sugieren que las redes LSTM generalizan mejor las dependencias a largo plazo, durante la realización de este trabajo surgieron GPT-3 (Brown et al., 2020) y Facebook Blender (Roller et al., 2020), sistemas Transformer que demuestran el potencial de esta arquitectura cuando tiene un tamaño de miles de millones de parámetros. Cabe pensar que su desempeño en este trabajo hubiera sido mejor de haber elegido *BERT_{large}* en lugar de *BERT_{base}*. Asimismo, además de aumentar el tamaño del

modelo, no hay que olvidar que el corpus era de textos en español; dada la complejidad de la conjugación verbal de este idioma, se necesitaría un número mayor de palabras en el vocabulario. No obstante, la cantidad de medios necesarios para entrenar estos modelos superaban los propósitos de este trabajo; el entrenamiento empleando modelos más grandes se deja, por tanto, como una investigación futura.

Otro elemento fundamental del desempeño insuficiente del modelo es la diferencia que existe entre el codificador ya preentrenado y el decodificador sin entrenar. Un trabajo futuro podría ser como indican, Liu and Lapata (2019), probar un entrenamiento del decodificador en dos etapas. En esta línea, Hoang et al. (2019) manifiesta la inefficiencia del uso de BERT para resumen, y propone volver al modelo autorregresivo en lugar del bidireccional de BERT.

En conclusión, el resultado más positivo de este trabajo es el conjunto de textos etiquetados. Se pueden aplicar a una infinidad de tareas de Procesamiento de Lenguaje Natural en español. Trabajos futuros podrían ser su utilización en clasificación de textos en español, desambiguación de autor o datación automática.

Otro resultado de interés ha sido el sistema de generación de argumentos inventados, que demuestra la capacidad lingüística y creativa de estos modelos, así como podría servir de ayuda a guionistas, novelistas o editoriales. Se han difundido sus resultados mediante el bot interactivo Apocrifosbot (<https://twitter.com/apocrifosbot>). BERT es capaz de adquirir las reglas del lenguaje con mayor precisión cuando la entrada se limita al tamaño para el que ha sido preentrenado. Así, el Experimento 2 parece confirmar, también, la investigación de Wu and Dredze (2019), quienes llaman la atención sobre la efectividad interlingüística de BERT.

En cuanto a la arquitectura Transformer en general, consideramos que su potencial está por explorar y este trabajo es, por lo tanto, una primera tentativa a nuevas e interesantes investigaciones de su aplicación al español.

Apéndice A

Palabras frecuentes por categoría



Figura A.1: Palabras más frecuentes para cada categoría



Figura A.2: Palabras más frecuentes para cada categoría



Figura A.3: Palabras más frecuentes para cada categoría

Bibliografía

- Alammar, J. (2019). The illustrated transformer. *Alammar GitHub Blog*.
- Allen, C. and Hospedales, T. (2019). Analogies explained: Towards understanding word embeddings. *arXiv preprint arXiv:1901.09813*.
- Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.
- Bengio, S. and Bengio, Y. (2000). Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- Cañete, J., Chaperon, G., Fuentes, R., and Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. In *to appear in PML4DC at ICLR 2020*.
- Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

- Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hoang, A., Bosselut, A., Celikyilmaz, A., and Choi, Y. (2019). Efficient adaptation of pretrained transformers for abstractive summarization. *arXiv preprint arXiv:1906.00138*.
- Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hornik, K., Stinchcombe, M., White, H., et al. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.
- Hsu, W.-T., Lin, C.-K., Lee, M.-Y., Min, K., Tang, J., and Sun, M. (2018). A unified model for extractive and abstractive summarization using inconsistency loss. *arXiv preprint arXiv:1805.06266*.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., and Klein, M. (2002). *Logistic regression*. Springer.
- Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing EMNLP-IJCNLP*. Association for Computational Linguistics.
- Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

- Lin, C. (2005). Recall-oriented understudy for gisting evaluation: Rouge. *Retrieved August, 20:2005.*
- Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Liu, Y. (2019). Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318.*
- Liu, Y. and Lapata, M. (2019). Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164.*
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*
- Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.
- Nenkova, A. and McKeown, K. (2011). *Automatic summarization*. Now Publishers Inc.
- Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.
- Pappagari, R., Źelasko, P., Villalba, J., Carmiel, Y., and Dehak, N. (2019). Hierarchical transformers for long document classification. *arXiv preprint arXiv:1910.10781.*
- Paulus, R. (2019). Deep reinforced model for abstractive summarization. US Patent 10,474,709.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365.*
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683.*

- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Ramos, J. et al. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ.
- Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Shuster, K., Smith, E. M., et al. (2020). Recipes for building an open-domain chatbot. *arXiv preprint arXiv:2004.13637*.
- Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.
- See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.
- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Tran, K., Bisazza, A., and Monz, C. (2018). The importance of being recurrent for modeling hierarchical structure. *arXiv preprint arXiv:1803.03585*.
- Treisman, A. M. and Gelade, G. (1980). A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

- Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.
- Wu, S. and Dredze, M. (2019). Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.
- Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.
- Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., and Lin, J. (2019). End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.
- Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2020a). Dive into deep learning. *Online*, page 993.
- Zhang, H., Gong, Y., Yan, Y., Duan, N., Xu, J., Wang, J., Gong, M., and Zhou, M. (2019a). Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*.
- Zhang, R., Wei, Z., Shi, Y., and Chen, Y. (2020b). {BERT}-{}al: {BERT} for arbitrarily long document understanding.
- Zhang, X., Wei, F., and Zhou, M. (2019b). Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*.
- Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.