UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DETECCIÓN AUTOMÁTICA DE PARÁFRASIS MEDIANTE EL MODELO SENTENCE-CROBI, UNA ARQUITECTURA DE RED NEURONAL SIMPLE BASADA EN CODIFICADORES CRUZADOS Y BI-CODIFICADORES

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
JESUS GERMAN ORTIZ BARAJAS

Directora de tesis:
DRA. GEMMA BEL ENGUIX
INSTITUTO DE INGENIERÍA, UNAM.

CIUDAD UNIVERSITARIA, CDMX. MAYO 2023

Las máquinas me toman por sorpresa
con mucha frecuencia.

Alan Turing

Agradecimientos

La realización de esta tesis se llevó a cabo gracias a la beca otorgada por el Consejo Nacional de Ciencia y Tecnología (CONACYT) que me permitió cursar mis estudios de maestría; al proyecto con clave TA101722 del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT); al proyecto CB A1-S-27780 del CONACYT; y a la Plataforma de Aprendizaje Profundo para Tecnologías del Lenguaje del Laboratorio de Súper-cómputo del Instituto Nacional de Astrofísica, Óptica y Electrónica.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

De acuerdo con Bhagat and Hovy (2013) la paráfrasis se refiere a enunciados que tienen el mismo significado pero que usan palabras diferentes para su redacción. El desarrollo de sistemas computacionales que puedan identificar la paráfrasis de forma automática es complejo debido a que definir que constituye un paráfrasis es complicado. Mota Montoya et al. (2016) señalan que algunos trabajos previos en el área definen la paráfrasis como una equivalencia aproximada entre los textos, además indican que existen diferentes tipos de paráfrasis con base en el nivel de cambios que haya en un texto respecto a otro. El primero es la paráfrasis baja, que consiste en la sustitución de sinónimos, hiperónimos, hipónimos, merónimos y holónimos. El segundo tipo es la paráfrasis alta, que además de la paráfrasis baja incluye cambios a nivel morfológico, léxico, semántico, sintáctico y discursivo. Debido al escenario anterior, una opción para enfrentar la identificación automática de paráfrasis es desarrollar soluciones basadas en aprendizaje profundo que nos permitan identificar los distintos tipos de paráfrasis sin la necesidad de extraer características lingüísticas complejas para definir los pares de textos.

En el ámbito del aprendizaje profundo, la arquitectura Transformer (Vaswani et al., 2017) introdujo una nueva era del procesamiento del lenguaje natural (PLN) con el surgimiento de los modelos del lenguaje pre-entrenados. Con el pre-entrenamiento, se pueden generar modelos que aprenden una representación a nivel general del lenguaje que pueden ajustarse a tareas especificas sin necesidad de entrenar un modelo desde cero para cada una de estas.

Humeau et al. (2019) explican que el modelo de clasificador cruzado es uno de los enfoques más populares basados en modelos de lenguaje pre-entrenados. Este modelo puede codificar dos textos de forma conjunta y aplica capas de auto-atención completa a ambos textos a la vez.

Por otro lado, Peng et al. (2022) explican que los bi-codificadores son otro enfoque basado en modelos del lenguaje pre-entrenados, el cual aplica la auto-atención a cada texto de manera separada utilizando una red neuronal siamesa (Bromley et al., 1993) y las compara a través de una métrica de similitud.

A partir del surgimiento del modelo BERT (Devlin et al., 2019), se han desarrollado múltiples enfoques para mejorar su rendimiento, dichos trabajos se basan en modificar la etapa de pre-entrenamiento, modificar el mecanismo de atención, destilación de conocimiento, entre otros aspectos complejos. En esta tesis se propone la arquitectura

Sentence-CROBI, una arquitectura simple que combina las representaciones obtenidas por codificadores cruzados y bi-codificadores para tareas de pares de textos, específicamente la identificación automática de paráfrasis.

1.1. Definición del problema

El problema a resolver es la identificación automática de paráfrasis. En esta tarea se recibe como entrada un par de enunciados y se debe determinar si son semánticamente equivalentes. Es decir, es un problema de clasificación binaria en el que a partir de dos textos S_1 y S_2 se debe determinar si son iguales o no.

1.2. Objetivo

El objetivo general de esta tesis es desarrollar una arquitectura de red neuronal basada en modelos del lenguaje pre-entrenados para resolver la tarea de identificación automática de paráfrasis. Los objetivos particulares son:

- Proponer una arquitectura de red neuronal simple basada en modelos del lenguaje pre-entrenados que obtenga resultados competitivos con el estado del arte.
- Comprobar su efectividad en los conjuntos de datos más utilizados en la tarea de identificación automática de paráfrasis.
- Ofrecer una alternativa atractiva para tareas a nivel de pares de textos fácil de implementar y con alta capacidad de adaptación.

1.3. Contribución

La principal contribución de esta tesis es la propuesta del modelo Sentence-CROBI. Es una arquitectura de red neuronal simple basada en modelos de lenguaje pre-entrenados que combina las representaciones vectoriales de los codificadores cruzados y los bi-codificadores para tareas de pares de textos. Para cuantificar el rendimiento de la arquitectura propuesta se evaluó en la tarea de identificación de paráfrasis.

Para maximizar su rendimiento se desarrollan diversos experimentos. En la primer ronda, se prueban diferentes técnicas de obtención de vectores contextuales de palabras en la componente siamesa de la red y se utilizan diferentes funciones de pérdida. Posteriormente, en la segunda ronda se prueban diferentes bloques de clasificación en los que varía la complejidad del modelo así como la función de *Pooling* óptima para generar las mejores representaciones vectoriales de cada texto. Finalmente, con base en los resultados en los tres conjuntos de datos utilizados y su comparación con el estado del arte, se exponen las ventajas de utilizar esta arquitectura.

1.4. Organización de la tesis

El resto de este trabajo se organiza de la siguiente manera: En el capítulo 2 se realiza un recorrido por los enfoques previamente desarrollados basados en modelos de lenguaje pre-entrenados utilizando la arquitectura *Transformer* a partir del surgimiento del modelo BERT. Estos modelos constituyen el estado del arte en diferentes tareas de procesamiento de lenguaje natural y son evaluados mediante el *GLUE Benchmark*, que es una colección de recursos para entrenar, evaluar y analizar sistemas de comprensión del lenguaje natural. Consiste en nueve tareas a nivel de enunciados o pares de enunciados, dentro de las que se encuentra la identificación de paráfrasis. Para facilitar la comprensión de los trabajos previos, se dividen en 4 diferentes ejes de investigación con base en las aportaciones de cada uno de ellos. Finalmente, se aborda el modelo de bi-codificadores, el cual es un componente de la arquitectura propuesta.

En el capítulo 3 se describen los conjuntos de datos utilizados para evaluar el rendimiento de la arquitectura propuesta, señalando las características particulares de cada uno de ellos, así como también se introduce un conjunto adicional que fue de utilidad para aumentar la eficacia del modelo.

Posteriormente, el capítulo 4 describe la metodología seguida para desarrollar esta tesis, desde el preprocesamiento del texto, la descripción detallada del modelo propuesto y cada uno de sus componentes, las estrategias de entrenamiento seguidas y las métricas de evaluación.

Dentro del capítulo 5 se presentan los resultados de todos los experimentos realizados tanto en la fase exploratoria como la versión final de la arquitectura desarrollada, así como las comparaciones y pruebas entre ellos. Finalmente, en el capítulo 6 se abordan las conclusiones de esta tesis y se mencionan algunas direcciones para posible trabajo futuro.

Capítulo 2

Trabajo relacionado

En este capítulo se presentan los enfoques previos basados en modelos de lenguaje pre-entrenados desarrollados a partir del surgimiento del modelo BERT (Devlin et al., 2019) en las sección 2.1. Estos trabajos son evaluados utilizando el *GLUE Benchmark*, que consta de 9 conjuntos de datos correspondientes a diferentes tareas de entendimiento de lenguaje natural. Dentro de estas tareas se encuentra la identificación de paráfrasis, la cual se evalúa con el corpus de paráfrasis de Microsoft (MRPC) y el conjunto de preguntas duplicadas de Quora (QQP). Los modelos descritos en esta sección fueron seleccionados con base en su rendimiento en el conjunto MRPC medido con la métrica *F1-score*. Los resultados fueron consultados en la tabla de posiciones del *GLUE Benchmark* disponible públicamente en línea[1].

Además, para facilitar la comprensión de los enfoques previos, estos se dividen en cuatro ejes de investigación: modificaciones al pre-entrenamiento del modelo del lenguaje, reducción al tamaño del modelo de lenguaje, modificación a la etapa de ajuste fino y modificaciones a la arquitectura Transformer. Finalmente, se describe el uso de modelos de lenguaje como bi-codificadores. Que es uno de los componentes de la arquitectura propuesta.

Las redes Transformer (Vaswani et al., 2017) son una arquitectura que permite codificar textos de manera paralela mediante el uso de mecanismos de atención en lugar de hacerlo de manera secuencial como las redes neuronales recurrentes. Esto permitió entrenar modelos con grandes cantidades de texto de forma eficiente, dando origen a los grandes modelos del lenguaje pre-entrenados que se han utilizado para resolver diversas tareas de procesamiento de lenguaje natural (Qiu et al., 2020; Markowitz, 2021).

2.1. Modelo BERT

BERT (Devlin et al., 2019) es el modelo de lenguaje basado en la arquitectura Transformer más conocido además de que obtuvo resultados del estado del arte en numerosas tareas (Rogers et al., 2020). Consta de dos versiones, *base* y *large* compuestas por 12 y 24 bloques de codificadores Transformer, respectivamente. En primer lugar, el modelo es

[1] `https://gluebenchmark.com/leaderboard`

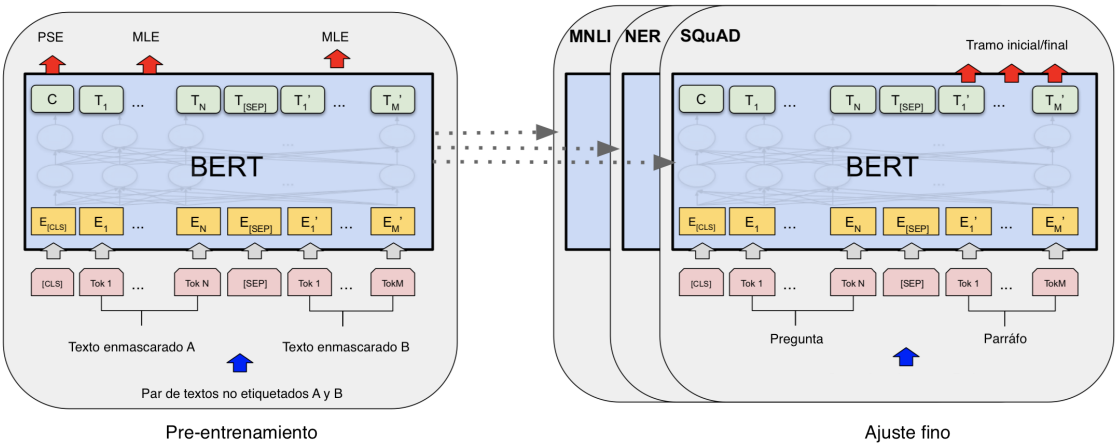


Figura 2.1: Diagrama general del proceso de pre-entrenamiento y ajuste fino del modelo BERT. Imagen extraída de Devlin et al. (2019).

pre-entrenado para aprender una representación general del lenguaje utilizando dos conjuntos de datos; el *BooksCorpus*, de 800 millones de palabras, y la Wikipedia en inglés, con 2,500 millones de palabras. El pre-entrenamiento del modelo se realiza mediante dos tareas. La primera es el modelo de lenguaje enmascarado, en la que una porción de los *tokens* de entrada se remplazan por la etiqueta [MASK] y el modelo aprende a predecir el valor real de dichos *tokens*; y la segunda es la predicción del siguiente enunciado en la que a partir de dos textos A y B el modelo debe identificar si B es el texto que procede de A o no. Después del pre-entrenamiento, se puede realizar un ajuste fino del modelo para resolver cualquier tarea de procesamiento del lenguaje natural agregando una capa al final del modelo, utilizando un número pequeño de épocas y una tasa de aprendizaje baja para ajustar todos los pesos de la arquitectura. La figura 2.1 muestra un diagrama general del proceso de pre-entrenamiento y ajuste fino del modelo BERT, como se observa, el único cambio en el modelo son las capas adicionales de cada tarea objetivo y es a partir de los pesos aprendidos durante el pre-entrenamiento que se inicializa el modelo para realizar el ajuste fino.

Posterior al surgimiento del modelo BERT, surgieron distintos enfoques que buscan mejorar su rendimiento con base en el mismo enfoque de pre-entrenamiento y ajuste fino de modelos basados en la arquitectura Transformer. Dichos enfoques se pueden dividir en cuatro ejes distintos, los cuales se describen en las subsecciones posteriores además de que se explican brevemente los enfoques propuestos en cada una de ellas.

2.1.1. Modificaciones al pre-entrenamiento del modelo del lenguaje

El primero consiste en la modificación de la etapa de pre-entrenamiento. El modelo RoBERTa (Liu et al., 2019), fue propuesto como una configuración optimizada de BERT; por lo que utiliza la misma arquitectura y las mismas variantes de tamaño *base* y *large*, con 12 y 24 bloques *Transformer*, respectivamente. En este enfoque, el enmascaramiento

de los *tokens* para la tarea de modelo de lenguaje enmascarado es dinámico, es decir, en cada inicio de época se genera un patrón de enmascaramiento diferente para los *tokens* del texto de entrada; que es crucial para pre-entrenar el modelo por más tiempo y con conjuntos de datos más grandes. Las demás modificaciones consisten en eliminar la tarea de predicción del siguiente enunciado, utilizar lotes más grandes y secuencias de texto más largas, además de emplear un conjunto de datos más extenso formado por los corpus *BookCorpus* (Zhu et al., 2015), *CC-News* (Hamborg et al., 2017), *OpenWebText* (Gokaslan and Cohen, 2019), y *Stories* (Trinh and Le, 2018).

En este mismo eje, el modelo StructBERT (Wang et al., 2020) fue propuesto como una extensión de BERT (Devlin et al., 2019) que incorpora información estructural del lenguaje al pre-entrenamiento del modelo del lenguaje mediante el ordenamiento a nivel de palabras y a nivel de enunciados, que permiten capturar de forma explicita las dependencias entre las palabras y enunciados en los vectores contextuales. Para la tarea de ordenamiento de palabras, se enmascara el 15 % de los *tokens* de la secuencia de entrada y posteriormente, se selecciona un cierto porcentaje de los *tokens* y se cambia el orden de forma aleatoria. Una vez obtenidos los vectores de salida de los bloques *Transformer*, estos se introducen a un clasificador Softmax para predecir el orden correcto de los *tokens*. La figura 2.2(a) ilustra el proceso de aprendizaje de la tarea de reordenamiento de palabras en conjunto con la tarea de modelo de lenguaje enmascarado, como se observa, se utilizan trigramas de *tokens* ($K = 3$) no enmascarados para mezclarlos alterando su orden, posteriormente los vectores h_i^L sirven como entrada a un clasificador para predecir el valor real de los *tokens*. Después se predice el orden correcto de los *tokens*, que equivale a maximizar la probabilidad de colocar cada *token* en su posición correcta como se denota en la ecuación 2.1. Donde:

- θ corresponde a los parámetros entrenables del modelo StructBERT.
- K representa la longitud de las subsecuencias mezcladas.

$$argmax_\theta \sum \log P(pos_1 = t_1, pos_2 = t_2, \ldots, pos_k = t_k | t_1, t_2, \ldots, t_k, \theta) \tag{2.1}$$

Para el ordenamiento a nivel de enunciados, dado un par de textos S_1 y S_2, el modelo se entrena para aprender a predecir si S_2 es el texto que sigue de S_1, si S_2 es el texto que precede a S_1 o si corresponde a un texto aleatorio de otro documento. Como se observa en la figura 2.2(b), en $\frac{1}{3}$ de los ejemplos S_2 corresponde al texto que sigue de S_1, $\frac{1}{3}$ de los casos S_2 es el texto anterior a S_1 y el $\frac{1}{3}$ restante corresponde a un texto aleatorio tomado de otro documento. Al igual que en BERT (Devlin et al., 2019), los textos son codificados de forma conjunta agregando el *token* [SEP] y el *token* de clasificación [CLS] es la entrada a un clasificador para asignar una de las tres clases posibles.

Para llevar a cabo el proceso de pre-entrenamiento se utilizan los conjuntos *BookCorpus* y la *Wikipedia* en inglés de forma análoga al procedimiento con BERT.

El último ejemplo dentro de este eje es el modelo Ernie 2.0 (Sun et al., 2020) en el que los autores proponen un marco de trabajo de aprendizaje multi-tarea continuo para aprender información léxica, sintáctica y semántica. Este marco de trabajo permite utilizar el conocimiento adquirido en las tareas anteriores para adaptarlo en las nuevas

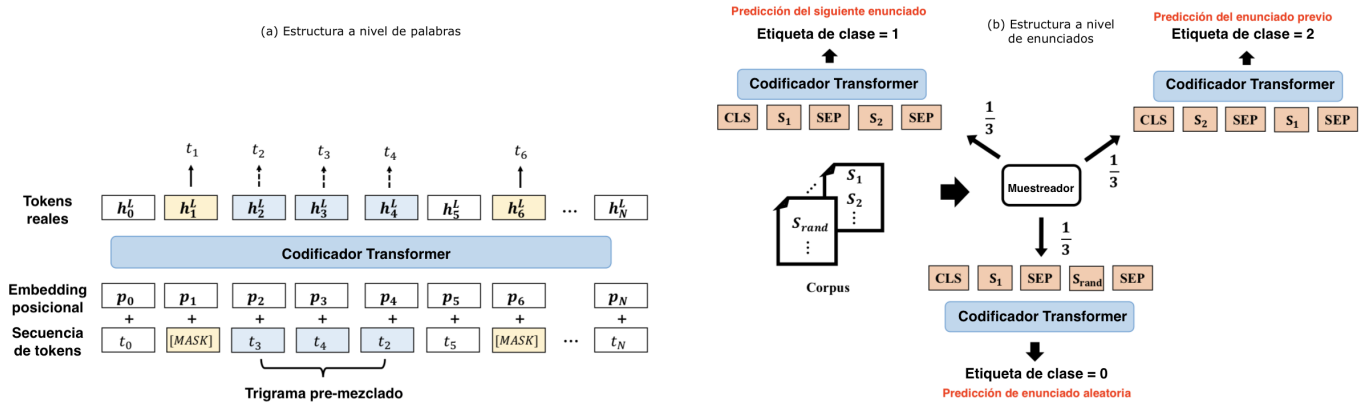


Figura 2.2: Diagramas de las tareas de ordenamiento a nivel de palabras y enunciados introducidas al modelo StructBERT. Imagen extraída de Wang et al. (2020).

tareas de pre-entrenamiento, por lo que cuando se introduce una nueva tarea, el modelo se entrena para aprenderla en conjunto con las tareas anteriores, utilizando el mismo bloque codificador en cada una de ellas. Para comprobar la eficacia del enfoque propuesto, los autores proponen un conjunto de siete tareas de pre-entrenamiento divididas en tres conjuntos como se muestra en la figura 2.3.

El primer conjunto consiste en tareas a nivel de palabras, que permite al modelo capturar información léxica de los corpus. La primera de ellas es el enmascaramiento de conocimiento, en el que de forma análoga a la tarea del modelo de lenguaje enmascarado, una fracción de las entidades nombradas y frases del texto son remplazadas por el *token* [MASK] y el modelo aprende a predecir el valor real de dichos *tokens*. La segunda tarea es la denominada relación *token*-documento, en la que el sistema aprende a predecir si el *token* de un segmento de texto aparece en otros segmentos del documento original. De forma empírica, las palabras que aparecen frecuentemente en diferentes partes de un documento son relevantes con el tópico del mismo, por lo que el modelo puede aprender a identificar palabras relevantes de los textos. La tercer tarea de este grupo es la predicción de mayúscula, en la que el modelo aprende a identificar si una palabra comienza con mayúscula o no, ya que de acuerdo con los autores, las palabras que inician con mayúsculas contienen cierta información semántica en comparación con otras palabras dentro del texto.

El segundo conjunto consta de tareas a nivel estructural para capturar la información sintáctica del corpus de pre-entrenamiento. En primer lugar, se tiene la tarea de reordenamiento del texto, en la que un párrafo dado se divide en m segmentos y son mezclados aleatoriamente. El modelo debe aprender a predecir la k-clase de cada segmento, tal que $k = \sum_{n=1}^{m} n!$, aprendiendo así las relaciones entre enunciados en un documento. La segunda tarea de este conjunto es la distancia entre textos, que es un problema de clasificación multi-clase en el que dados dos textos se debe determinar si son textos adyacentes en el mismo documento, si están en el mismo documento pero no son adyacentes y si no pertenecen al mismo documento.

El último conjunto es de tareas a nivel semántico. La primer tarea es la relación del discurso, en la que el modelo aprende a predecir la relación semántica o retórica de dos textos. La segunda es la relevancia en un sistema de recuperación de información, que está

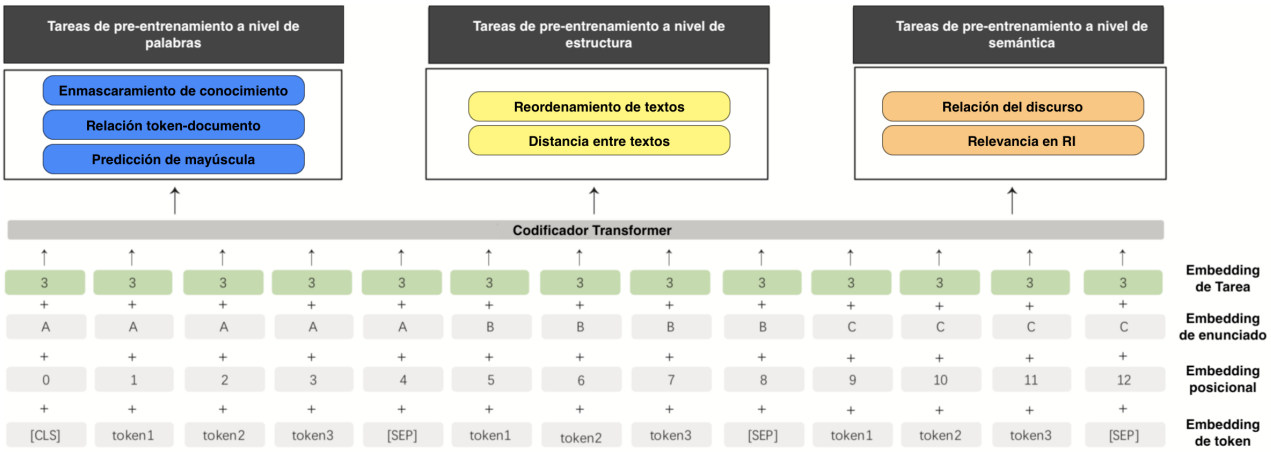


Figura 2.3: Estructura del modelo Ernie 2.0. Imagen extraída de Sun et al. (2020).

propuesta como una problema de clasificación multi-clase con tres etiquetas, en la que dado una consulta, compuesta por el primer texto, y un título, compuesto por el segundo texto, el modelo debe predecir si hay una relevancia fuerte, débil o nula.

2.1.2. Reducciones al tamaño del modelo del lenguaje

El segundo eje de modificaciones consiste en reducir el tamaño de los modelos. El modelo ALBERT (Lan et al., 2020) fue propuesto como una alternativa para ser utilizada en entornos con memorias de Unidades Gráficas de Procesamiento (GPU, por sus siglas en inglés) y Unidades de Procesamiento de Tensores (TPU, por sus siglas en inglés) limitadas. Los autores proponen dos técnicas para reducir el tamaño del modelo BERT (Devlin et al., 2019). La primer técnica consiste en la compartición de parámetros entre los bloques Transformer del modelo, ya que al agregar más bloques a las versiones *base*, con 12 bloques Transformer, o la versión *large* con 24 bloques, el número de parámetros del modelo crece de forma exponencial, como se muestra en la tabla 2.1, donde se observa una diferencia de 230 millones de parámetros entre las distintas versiones de BERT (Devlin et al., 2019). La compartición de parámetros permite que en lugar de aprender pesos únicos para cada bloque Transformer, ya sean 12 o 24, se aprendan únicamente pesos para un bloque y este se reutilice las veces que sea necesario. Con esta técnica, los autores reducen de 110 millones de parámetros la versión *base* a únicamente 31 millones de parámetros utilizando los mismos bloques y el tamaño oculto de 768 unidades.

Versión	**Unidades ocultas**	**No. de capas**	**No. de parámetros**
BERT-large	1024	24	340 millones
BERT-base	768	12	110 millones

Tabla 2.1: Estadísticas de tamaño de las versiones *large* y *base* del modelo BERT (Devlin et al., 2019).

La segunda técnica es la parametrización de *embeddings* factorizada. Al descomponer

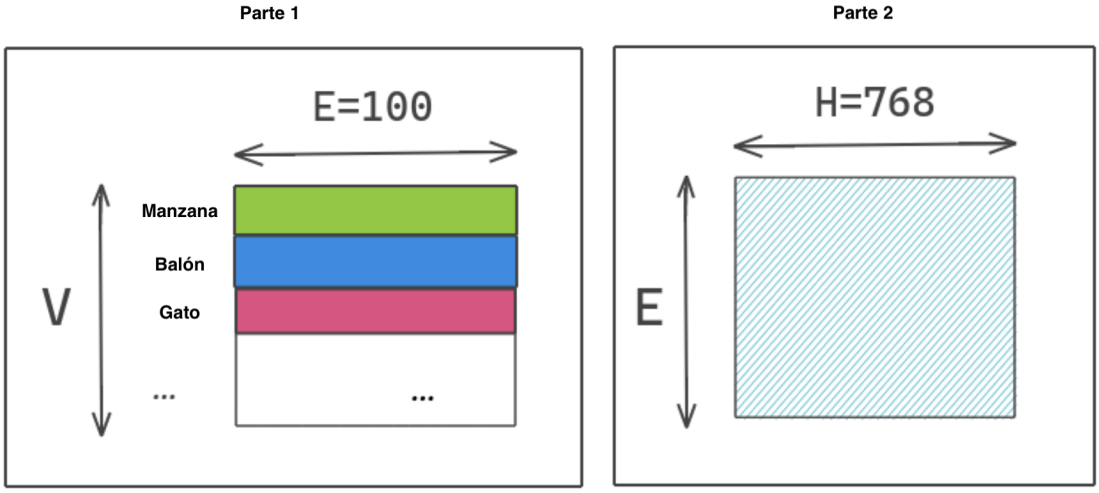


Figura 2.4: Parametrización de embeddings factorizada propuesta en el modelo ALBERT. Imagen extraída de Lan et al. (2020).

la matriz de *embeddings* del vocabulario en dos matrices más pequeñas permite separar el tamaño de las capas ocultas del modelo del tamaño del tamaño del vocabulario, lo que facilita el crecimiento del tamaño de las representaciones ocultas sin afectar significativamente al tamaño del vocabulario. En el modelo BERT (Devlin et al., 2019), el tamaño de los embeddings de entrada es igual al tamaño de las representaciones ocultas del modelo, esto es $E = H = 768$, para el caso *base*; de forma que si se aumenta el tamaño de las representaciones ocultas del modelo, entonces es necesario agregar una dimensión a los embeddings del modelo, que tiene un tamaño de $V \times E$ y resulta en modelos con millones de parámetros. Como se muestra en la figura 2.4, la parametrización factorizada consiste en proyectar los vectores de entrada en un espacio vectorial E de menor dimensión, en este caso $E = 100$, y posteriormente proyectarlos sobre el espacio vectorial de las representaciones ocultas.

Siguiendo con el eje de reducción del tamaño, el modelo BORT (de Wynter and Perry, 2020) es presentado como una sub-arquitectura óptima de BERT obtenida a través de un esquema de aproximación de tiempo completamente polinomial (FPTAS, por sus siglas en inglés) con base en tres métricas de evaluación: tiempo de inferencia, tamaño del modelo y tasa de error. Para encontrar la sub-arquitectura óptima se utilizó el espacio de búsqueda $\Xi = D \times A \times H \times I$ siguiente:

- Bloques Transformer de codificación $D = \{2, 4, 6, 8, 10, 12\}$
- Cabezas de atención $A = \{4, 8, 12, 16\}$
- Tamaño del espacio oculto $H = \{512, 768, 1024\}$
- Tamaño de capas intermedias $I = \{256, 512, 768, 1024, 3072\}$

La sub-arquitectura con mejor rendimiento tiene los parámetros $\{4, 8, 1024, 768\}$ que corresponden a 4 bloques Transformer de codificación, 8 cabezas de atención, 1024 en

el tamaño del espacio oculto y 768 en el tamaño de las capas intermedias; dando como resultado un modelo con únicamente 57 millones de parámetros entrenables, que representa una diferencia de 53 millones de parámetros con respecto al modelo BERT *base* (Devlin et al., 2019).

Sin embargo, al tratarse de una arquitectura 50 % y 83 % más pequeña que BERT *base* y BERT *large*, respectivamente, es más propensa al sobreajuste, por lo que los autores utilizan el algoritmo codicioso Ágora (de Wynter, 2020) para la etapa de ajuste fino, el cual combina técnicas de aumento de datos y destilación de conocimiento. De forma general, el algoritmo utiliza dos modelos: Timeo y Sócrates. En la $k-$ésima iteración, el modelo Timeo se entrena sobre un conjunto de datos D^k con todas las combinaciones de hiperparámetros en Θ^k y se selecciona el modelo con mejor rendimiento con base en la Exactitud o *Accuracy*. Posteriormente se crea un conjunto M^k con nuevos ejemplos de entrenamiento, se etiquetan usando el modelo Sócrates, que es un modelo pre-entrenado y sobre el cual se realizó un proceso de ajuste fino en la tarea que se está entrenando el modelo Timeo, se agregan al conjunto de entrenamiento D^{k+1} y se actualiza el conjunto de hiperparámetros Θ^{k+1}.

2.1.3. Modificaciones al algoritmo de ajuste fino

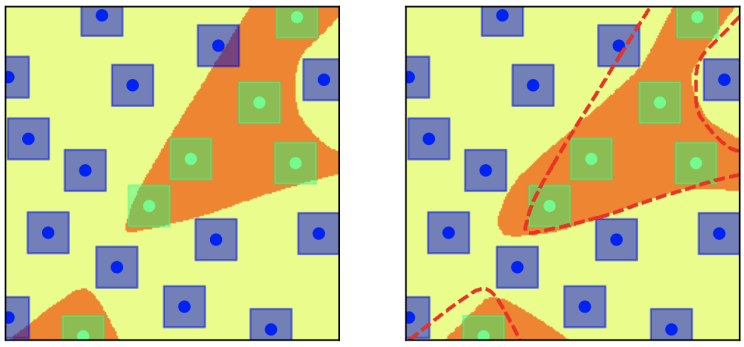
El tercer eje consiste en modificar la etapa de ajuste fino para lograr mejorar la capacidad de generalización de los modelos del lenguaje. Debido a la limitada cantidad de datos en las tareas objetivo y a la gran complejidad de las arquitecturas propuestas en el estado del arte, los modelos del lenguaje son propensos a sobreajustarse a los ejemplos de entrenamiento. Dado el problema anterior, se propuso el algoritmo *SMART* (Jiang et al., 2020), compuesto por dos ingredientes principales. El primero es una técnica de regularización adversaria inductora de suavidad con la que a través de agregar una leve perturbación a los ejemplos de entrada se logra controlar la capacidad del modelo y su alta complejidad. De forma forma general, el método consiste en minimizar la ecuación 2.2 durante el ajuste fino del modelo:

$$\min_{\theta} F(\theta) = L(\theta) + \lambda_s R_s(\theta) \tag{2.2}$$

Donde:

- La función de error se define como $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i;\theta), y_i)$.
- λ_s es un hiperparámetro.
- El regularizador se define como $R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x_i}-x_i\|\leq\epsilon} l_s(f(\tilde{x_i};\theta), f(x_i;\theta))$

El regularizador adversario permite que la salida del modelo no cambie mucho cuando se agrega una pequeña perturbación, por lo que al minimizar la ecuación 2.2 provoca que el modelo sea uniforme dentro de todas las vecindades de las entradas x_i. La figura 2.5(a) muestra los límites de la región de decisión aprendida por el modelo cuando no se utiliza la regularización y su comparación cuando se utiliza la regularización con base en el algoritmo



(a) (b)

Figura 2.5: Límites de decisión aprendidos sin regularización (a) y con regularización (b), propuesta por en el algoritmo SMART. Imagen extraída de Jiang et al. (2020).

SMART (Jiang et al., 2020) en 2.5(b). Es posible observar cómo la salida f no cambia mucho dentro de la vecindad de los puntos de entrenamiento.

El segundo ingrediente del algoritmo SMART (Jiang et al., 2020) es un método de optimización para la ecuación 2.2 basado en los métodos de Bregman, cuya idea es imponer una penalización en cada iteración para evitar la actualización agresiva de los parámetros del modelo. Tomando como punto de partida un modelo pre-entrenado $f(\cdot;\theta)$, en la iteración $t+1$ el método de Bregman toma la ecuación 2.3.

$$\theta_{t+1} = argmin_{\theta} F(\theta) + \mu D_{BREG}(\theta, \theta_t) \tag{2.3}$$

Donde:

- μ es un parámetro que se aprende durante el entrenamiento.
- La divergencia de Bregman se define como $D_{BREG}(\theta, \theta_t) = \frac{1}{n}\sum_{i=1}^{n} l_s(f(x_i;\theta), f(x_i;\theta_t))$

Dada la ecuación 2.3, cuando μ tiene un valor muy grande, la divergencia de Bregman funciona como un regularizador que previene que los valores de los pesos del modelo en θ_{t+1} se alejen mucho del valor de la iteración previa θ_t.

2.1.4. Modificaciones a la arquitectura Transformer

Finalmente, el cuarto eje consiste en modificaciones a la arquitectura Transformer. En el modelo DeBERTa (He et al., 2020), los autores proponen un cambio a la arquitectura al que denominan atención desenredada; en la cual las palabras son codificadas utilizando dos vectores, uno en el que se captura el contenido o significado y otro en el que se codifica la posición relativa dentro del texto. Este cambio permite resolver la limitante del modelo BERT Devlin et al. (2019), en el que los vectores de contenido y posición se suman para obtener la representación vectorial de un *token*, permitiendo así que una misma palabra tenga vectores distintos para posiciones diferentes. Debido a esta nueva codificación, los pesos de la atención se componen de tres matrices: contenido a contenido, contenido a posición y posición a contenido. Además para la etapa de pre-entrenamiento mediante la

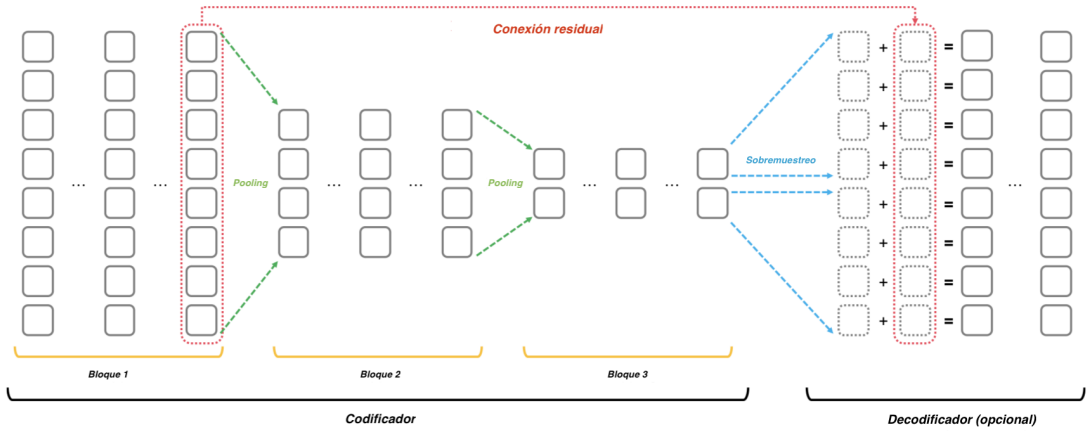


Figura 2.6: Diagrama de la arquitectura Funnel-Transformer. Imagen extraída de Dai et al. (2020).

tarea de modelado de lenguaje enmascarado, los autores añaden una vector que codifica la posición absoluta de los *tokens*, lo que evita ambigüedades en función del rol sintáctico de las palabras en el texto.

En el mismo ámbito, el modelo Funnel-Transformer (Dai et al., 2020) fue propuesto con el objetivo de reducir el costo computacional de pre-entrenar un modelo del lenguaje en un conjunto de datos muy grande para tareas en las que se requiere una representación vectorial simple de un texto o par de textos, en las que al mantener un tamaño fijo de representaciones ocultas a lo largo de la arquitectura puede provocar redundancia gracias a la granularidad a nivel de *tokens*. Por lo anterior, en este modelo se propone reducir gradualmente la resolución de las representaciones ocultas en los modelos de auto-atención, para ello se agrega una operación de *pooling* posterior a un bloque Transformer l del codificador para reducir el tamaño de las representaciones ocultas del texto de entrada, además con el objetivo de reconstruir una representación vectorial de la dimensión inicial en tareas a nivel de *tokens* como el enmascaramiento de *tokens* durante el pre-entrenamiento se emplea un decodificador, el cual puede ser descartado para tareas a nivel de enunciado cuando se realiza el ajuste fino del modelo. La figura 2.6 muestra un diagrama de alto nivel de la arquitectura Funnel-Transformer. El codificador está compuesto por varios bloques Transformer, en el que la representación oculta no cambia su tamaño; sino que es a partir de un bloque más profundo que se aplica la operación de *pooling* para reducir el tamaño de la representación oculta a la mitad. Además, los autores proponen una variante en la que el vector reducido por la operación de *pooling* se utiliza como el vector *query*, mientras que el vector no reducido se utiliza como vectores *key* y *value*, por lo que la reducción de la dimensión no depende únicamente de la operación de *pooling*, sino que además toma en cuenta cómo la auto-atención suma cada vector no reducido para formar la representación reducida. Esta variante sigue el principio lingüístico de que *tokens* contiguos pueden fusionarse gradualmente para crear componentes semánticos más grandes.

2.2. Modelos del lenguaje como bi-codificadores

A diferencia de los trabajos descritos anteriormente, existe otro enfoque basado en modelos de lenguaje basados en la arquitectura Transformer pre-entrenados llamado bi-codificadores, en los que para una tarea de pares de textos, cada par es codificado de manera independiente mediante una red neuronal siamesa (Bromley et al., 1993) y se obtiene una representación vectorial separada de cada par.

El modelo Sentence-BERT (Reimers and Gurevych, 2019) utiliza dos instancias del modelo BERT con pesos compartidos donde cada texto se codifica de forma independiente, a la salida del modelo BERT, se aplica una operación de *pooling* al último estado oculto del modelo para obtener los vectores de cada texto; estas representaciones se combinan para obtener una representación global. En la figura 2.7 se muestra el diagrama de la arquitectura Sentence-BERT para tareas de clasificación. Como se observa, los textos A y B son procesados de forma individual por dos instancias del modelo BERT con pesos compartidos, posteriormente, a la matriz del último estado oculto del modelo se le aplica una operación de *pooling* para obtener las representaciones vectoriales individuales de cada texto u y v, respectivamente. Finalmente, se concatenan los vectores u y v además de la diferencia absoluta entre estos vectores $|u - v|$ para obtener la representación global del par de enunciados, que sirve como entrada a una red completamente conectada que realiza la clasificación.

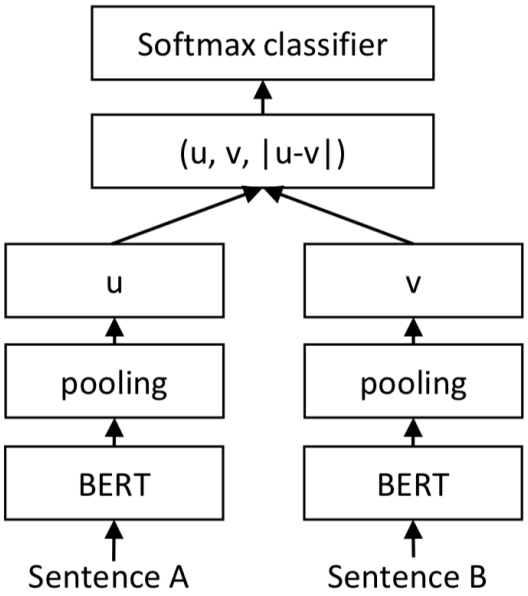


Figura 2.7: Diagrama de la arquitectura Sentence-BERT para tareas de clasificación. Imagen extraída de Reimers and Gurevych (2019).

2.3. Resumen

El surgimiento de las redes Transformer (Vaswani et al., 2017) dio paso a una nueva era en el campo de procesamiento de lenguaje natural gracias a su capacidad de entrenar de forma eficiente arquitecturas de redes neuronales con grandes cantidades de texto, dando origen a los grandes modelos del lenguaje pre-entrenados cuyo propósito es aprender representaciones generales del lenguaje y posteriormente ser adaptados a tareas específicas sin necesidad de crear un modelo diferente para cada una de estas, sino solamente agregar una capa adicional en función de la tarea a resolver. El modelo BERT (Devlin et al., 2019) es el modelo de lenguaje pre-entrenado más popular que sigue este enfoque de dos fases: pre-entrenamiento y ajuste fino, y que además alcanzó resultados del estado del arte en una amplia variedad de tareas. Desde su aparición, se ha desarrollado un campo de investigación que consiste en mejorar el rendimiento de este tipo de modelos que podemos separar en cuatro ejes principales: modificaciones al pre-entrenamiento de los modelos, reducciones al tamaño del modelo, modificar la etapa de ajuste fino y modificar la arquitectura Transformer (Vaswani et al., 2017). En esta tesis se presenta una arquitectura que combina dos tipos de representaciones basadas en modelos del lenguaje, codificadores cruzados y bi-codificadores de manera simple para la resolución de tareas de pares de texto, específicamente la identificación automática de paráfrasis.

Capítulo 3

Conjuntos de datos

En este capítulo se describen los conjuntos de datos utilizados para evaluar la arquitectura propuesta en la tarea de identificación automática de paráfrasis. La elección fue realizada con base en el sitio web *Papers with Code*[1], en el que se ofrece información acerca de los temas en tendencia en la investigación en aprendizaje de automático. Dentro del sitio es posible consultar los resultados históricos en diferentes tareas así como los conjuntos de datos empleados en las mismas. Por lo tanto, la selección de los conjuntos de datos para este trabajo se realizó con base en el número de citas que cada conjunto tenía para la tarea de detección automática de paráfrasis.

3.1. *Microsoft Research Paraphrase Corpus*

En primer lugar, se utiliza el corpus de paráfrasis de Microsoft (MRPC) propuesto por (Dolan and Brockett, 2005). Surge a partir de la necesidad de tener un conjunto de datos de gran escala y etiquetado con el cual sea posible desarrollar investigación en los temas de identificación y generación de paráfrasis, además de incentivar la publicación de conjuntos similares para promover la comparación de los distintos enfoques propuestos.

Los textos que componen el corpus fueron obtenidos de una base de datos de 13,127,938 pares de oraciones, extraídos de 9,516,684 textos de 32,408 grupos de noticias recopilados de internet en un periodo de dos años. En primer lugar, los autores recolectaron un conjunto de 49,375 pares que utilizaron para entrenar un clasificador basado en máquinas de vectores de soporte para detectar paráfrasis. Del conjunto inicial 20,574 pares fueron clasificados como paráfrasis, y es a partir de este subconjunto que se extrajeron aleatoriamente 5,801 pares de enunciados para ser evaluados manualmente por dos jueces, los cuales decidieron de forma independiente si eran paráfrasis o no. En caso de discrepancia en el etiquetado, un tercer juez daba su veredicto y la etiqueta final se asignó con base en un voto mayoritario.

El corpus está dividido en subconjuntos para entrenamiento y prueba. El conjunto de entrenamiento se compone de 4,076 pares de enunciados, los cuales 2,753 fueron etiquetados como semánticamente equivalentes, es decir, 67.5 % de los pares corresponden a la clase Paráfrasis, mientras que los 1,323 pares restantes de este conjunto fueron etiquetados como No paráfrasis.

[1] `https://paperswithcode.com`

Por otro lado, el conjunto de prueba está formado por 1,725 pares de enunciados, de los cuales 66.5 % fueron etiquetados como Paráfrasis, es decir, 1,147 pares de enunciados. Los 578 pares restantes fueron etiquetados como No paráfrasis.

3.2. *PAWS-Wiki*

El segundo conjunto de datos utilizado es el corpus *Paraphrase Adversaries from Word Scrambling* (*PAWS*). Desarrollado por (Zhang et al., 2019), este conjunto surge por la necesidad de tener ejemplos de pares de textos con alta superposición léxica sin ser semánticamente equivalentes. Es decir, ejemplos tales como:

1. Flights from New York to Florida.
2. Flights to Florida from NYC.
3. Flights from Florida to New York.

Como se observa en la lista anterior, los 3 enunciados poseen una alta superposición léxica, sin embargo, únicamente los enunciados 1 y 2 son Paráfrasis, mientras que el enunciado 3 tiene un significado diferente.

La figura 3.1 muestra el flujo seguido por los autores para la creación del corpus PAWS. La primer fase consiste en crear pares de enunciados a partir de un corpus original compuesto de textos de Quora y Wikipedia. Esta fase se basa en dos ideas; la primera de ellas consiste en cambiar el orden de las palabras con ayuda de un modelo del lenguaje y teniendo supervisión humana. Para realizar esto, se obtienen las partes de la oración (POS) del texto y se agrupan aquellas palabras y frases con la misma oración para crear los conjuntos candidatos que se usan para cambiar el orden de las palabras del enunciado original. A partir de una búsqueda de haz, se llenan los espacios del enunciado generado de izquierda a derecha con base en las etiquetas POS del texto original y los conjuntos candidatos. Finalmente, los textos generados son evaluados mediante un modelo del lenguaje.

La segunda idea de generación de textos se basa en la retro-traducción, que consiste en traducir un texto S en idioma A al idioma B y posteriormente traducirlo de nuevo al idioma A para así tener un texto S' semánticamente equivalente. Para llevar a cabo este proceso, los autores utilizaron dos modelos de traducción automática, el primero inglés-alemán y el segundo alemán-inglés. Después de obtener los textos retro-traducidos, se aplicaron una serie de heurísticas basadas en la bolsa de palabras del texto original para controlar la calidad de las paráfrasis generadas.

Finalmente se realiza el etiquetado final para todo par (s_1, s_2) generado por el reordenamiento de palabras y para todo par (s_1, s_1') generado por el proceso de retro-traducción con base en las siguientes regalas:

1. (s_2, s_1') son Paráfrasis si (s_1, s_2) y (s_1, s_1') son Paráfrasis.
2. (s_2, s_1') son No paráfrasis si alguno de (s_1, s_2) o (s_1, s_1') son No paráfrasis.
3. (s_2, s_1') no se incluye en cualquier otro caso debido a que su etiqueta es desconocida.

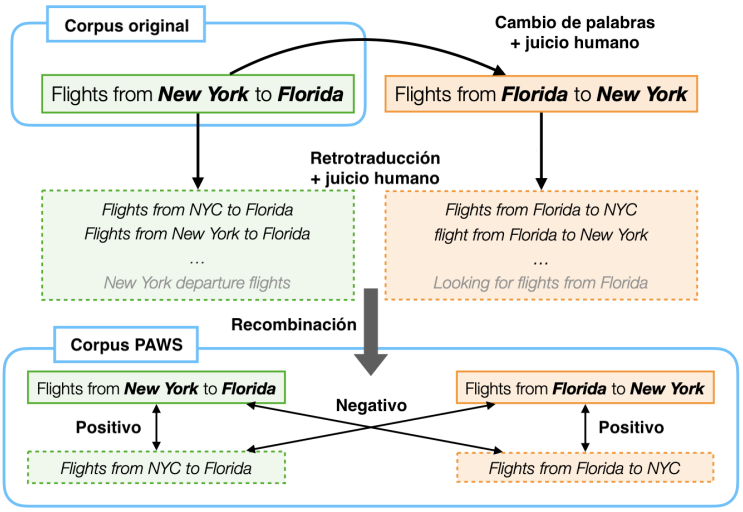


Figura 3.1: Diagrama del flujo de creación del corpus PAWS. Imagen extraída de Zhang et al. (2019).

Para esta tesis, se utilizó el subconjunto PAWS-Wiki, compuesto únicamente por los textos de Wikipedia. Consta de 65,401 pares de textos divididos en tres subconjuntos, de entrenamiento con 49,401 ejemplos y validación y prueba con 8,000 cada uno. La distribución del corpus es de 44 % de ejemplos etiquetados como Paráfrasis, y 56 % etiquetados como No paráfrasis.

3.3. *Quora Question Pairs*

El último conjunto de datos utilizado es el corpus de Preguntas Duplicadas de Quora[2] (QQP). Este conjunto de datos está conformado por 795,241 pares de preguntas etiquetadas de forma binaria como *Duplicada* o *No Duplicada*. Está dividido en tres subconjuntos; de entrenamiento, con 363,846 pares de preguntas, de validación con 40,430 y finalmente de prueba, con 390,965 pares de textos. Para los subconjuntos de entrenamiento y validación se tiene una distribución del 37 % para las preguntas duplicadas y 63 % para las preguntas no duplicadas; mientras que la distribución del conjunto de prueba es desconocido. Para asegurar la consistencia y validez de los resultados obtenidos con este conjunto de datos, en este trabajo se utilizaron los datos proporcionados por el *GLUE Benchmark* (Wang et al., 2019), un conjunto de tareas para evaluar el rendimiento de los sistemas en diversas tareas de entendimiento del lenguaje natural. A los autores del *GLUE Benchmark* se les proporcionó de forma privada las etiquetas del subconjunto de prueba y señalan que no posee la misma distribución que en los subconjuntos de entrenamiento y validación. Finalmente, es importante resaltar que para obtener los resultados de la arquitectura propuesta en este conjunto de datos, utilizamos el servidor público de la competencia[3].

La tabla 3.1 muestra las estadísticas de los conjuntos de datos descritos anteriormente.

[2] `https://www.quora.com/profile/Ricky-Riche-2/First-Quora-Dataset-Release-Question-Pairs`
[3] `https://gluebenchmark.com/submit`

Para cada corpus se describen sus particiones indicando cuántos pares pertenecen a cada clase.

Corpus	Instancias de Paráfrasis	Instancias de No paráfrasis	Instancias totales
MRPC (*train*)	2,753	1,323	4,076
MRPC (*test*)	1,147	578	1,725
QQP (*train*)	134,623	229,223	363,846
QQP (*val*)	14,959	25,471	40,430
QQP (*test*)	-	-	390,965
PAWS-Wiki (*train*)	21,829	27,572	49,401
PAWS-Wiki (*val*)	3,539	4,461	8,000
PAWS-Wiki (*test*)	3,536	4,464	8,000

Tabla 3.1: Distribución de los conjutos de datos utilizados: MRPC, QQP y PAWS-Wiki

3.4. Conjuntos de datos adicionales

Adicionalmente, para este trabajo se utilizó el conjunto *Multi-Genre Natural Language Inference* (MultiNLI) (Williams et al., 2018) propuesto para la tarea de inferencia del lenguaje natural. Está compuesto por pares de textos de diferentes dominios etiquetados en una de tres clases, estas indican la relación que existe entre los textos: neutral, contradicción y vinculación. El conjunto de datos está dividido en tres subconjuntos, de entrenamiento, validación y prueba. Siguiendo el enfoque de ajuste fino en dos fases propuesto por Phang et al. (2018) que se describirá con mayor detalle en el capítulo 4, se utilizan los subconjuntos de entrenamiento y validación para realizar un ajuste fino intermedio antes de ajustar el modelo a las tareas objetivo. El conjunto de entrenamiento consta de 391,164 ejemplos de los cuales 130,375 corresponden a la clase neutral, 130,379 son de la clase contradicción y 130,411 pertenecen a la clase vinculación. Por su parte, el conjunto de validación está formado por 9,714 ejemplos de los que 3,094 pares están etiquetados como neutral, la clase contradicción la conforman 3,180 pares y finalmente, la clase vinculación está formada por 3,440 instancias.

Capítulo 4

Metodología

En esta sección se explica detalladamente la metodología seguida para el desarrollo de esta tesis. En la sección 4.1 se aborda el pre-procesamiento realizado sobre el texto para preparar las entradas para la arquitectura propuesta. Posteriormente, la sección 4.3 presenta la arquitectura Sentence-CROBI y cada uno de sus componentes. Finalmente, las secciones 4.4, 4.6 y 4.8 abordan los algoritmos de ajuste fino del modelo, la técnica de aprendizaje conjunto utilizada y las métricas de evaluación, respectivamente.

4.1. Pre-procesamiento del texto

En esta sección se describe el pre-procesamiento realizado sobre los pares de texto para crear las entradas de la red Sentence-CROBI. Puesto que este modelo tiene como componentes un codificador cruzado y un bi-codificador, los pares de texto se deben codificar de forma conjunta y de manera individual. En el caso de este trabajo, utilizamos dos combinaciones diferentes, en la primera de ellas, utilizamos el modelo BERT-base (Devlin et al., 2019) tanto para el codificador cruzado como para el bi-codificador, mientras que en la segunda combinación se utiliza el modelo RoBERTa-large (Liu et al., 2019) como codificador cruzado y BERT-base (Devlin et al., 2019) como bi-codificador. Para el caso del modelo BERT (Devlin et al., 2019) el vocabulario está compuesto por 30,000 *tokens*, mientras que el vocabulario del modelo RoBERTa (Liu et al., 2019) consta de 50,000 *tokens*. De esta forma, el vocabulario del modelo BERT-base (Devlin et al., 2019) V_{BERT} se representa por una matriz de $n \times m$, donde $n = 30,000$ y $m = 768$, representan el número de *tokens* y las dimensiones de cada vector, respectivamente. El pre-procesamiento realizado se describe a continuación:

1. Se *tokeniza* a nivel de palabras cada texto y se obtiene una secuencia de IDs de cada uno, que representa el renglón n_i de la matriz del vocabulario.

2. Para el caso de la codificación conjunta del par de textos se agrega el *token* especial de clasificación [CLS] al inicio y se separan ambas secuencias de IDs con el *token* especial de separación [SEP].

3. Para el caso de la codificación individual de cada texto se agrega el *token* especial de

clasificación [CLS] al inicio y se añade el *token* especial de separación [SEP] al final de cada texto.

4. Posteriormente, se agrega un relleno a ambas codificaciones para unificar el tamaño de las representaciones iniciales. Se asignó un tamaño limite de 128 *tokens* para el caso de los pares y de 35 para los textos individuales.

5. Finalmente, se obtienen las mascarás de atención para ambas representaciones, para que el modelo sea capaz de distinguir entre los *tokens* de palabras y los *tokens* del relleno.

4.2. Arquitectura *Transformer*

Las *Transformers* (Vaswani et al., 2017), son redes neuronales artificiales pre-alimentadas con un mecanismo de auto-atención que han tenido un gran éxito en tareas de procesamiento del lenguaje natural y en general en problemas que involucran el modelado de datos secuenciales. En esta sección se describe el funcionamiento de cada uno de los componentes de esta arquitectura con base en el trabajo de Phuong and Hutter (2022).

4.2.1. *Embedding* de *tokens*

En esta etapa la red *Transformer* aprende a representar cada elemento del vocabulario V como un vector en $\mathbb{R}^{d_e}$.

Dicho vocabulario V es un conjunto finito de elementos llamados *tokens*, los cuales pueden ser palabras o caracteres, no obstante, lo más común es utilizar sub-palabras, que permiten al modelo lidiar con nuevas palabras en tiempo de evaluación. Posterior a la elección de la técnica de tokenización, a cada *token* se le asigna un identificador o ID único y finalmente se añaden *tokens* especiales para representar el inicio y el fin de la secuencia, y en casos como el modelo BERT, un *token* de enmascaramiento.

Por lo tanto, este bloque del modelo tiene como entrada un ID v que representa una porción del texto y a la salida un vector $e = W_e[:, v]$ proveniente de la matriz de *embeddings* de *tokens* $W_e \in \mathbb{R}^{d_e \times N_v}$.

4.2.2. *Embedding* posicional

Durante esta fase, el modelo codifica la posición de un *token* en una secuencia como un vector en $\mathbb{R}^{d_e}$, cuyo propósito es permitir que la red tenga noción del orden de las palabras. Por ejemplo, la posición del primer *token* de la secuencia se representa con un vector $W_p[:, 1]$, el segundo *token* se representa por otro vector $W_p[:, 1]$, y así sucesivamente.

Para llevar a cabo esta codificación, es necesario que las secuencias de entrada tengan un tamaño fijo l_{max}. La propuesta de Vaswani et al. (2017) utiliza la ecuaciones 4.1 y 4.2 para aprender el vector de posición para $0 < i \leq d_e/2$.

$$W_p[2i-1, t] = sin(\frac{t}{l_{max}^{\frac{2i}{d_e}}}) \tag{4.1}$$

$$W_p[2i, t] = cos(\frac{t}{l_{max}^{\frac{2i}{de}}}) \tag{4.2}$$

Finalmente, el *embedding* posicional se suma al *embedding* de *token* para formar la representación vectorial inicial del *token* de entrada e como se muestra en la ecuación 4.3.

$$e = W_e[:, x] + W_p[:, t] \tag{4.3}$$

La Figura 4.1 muestra un diagrama de la representación vectorial inicial de la secuencia de entrada. Como se observa, cada *token* se representa por un *embedding* de dimensión 4. Posteriormente se calculan los *embeddings* de posición que se suman a los *embeddings* de *tokens* para formar los vectores de entrada a la red.

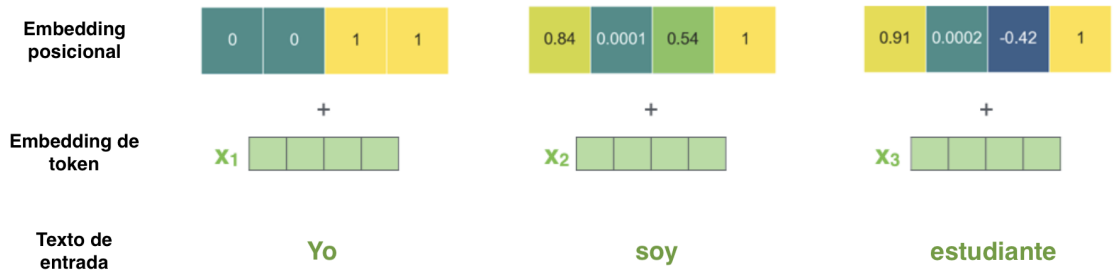


Figura 4.1: Representación vectorial inicial de la secuencia de entrada a la red Transformer. Imagen extraída de Alammar (2018).

4.2.3. Atención

La auto-atención es el componente principal de la arquitectura *Transformer*. Le permite a la red neuronal utilizar la información contextual, es decir, el texto anterior o el texto posterior, para predecir el *token* actual.

De forma general, la atención funciona de la siguiente forma: el *token* que se predice actualmente se asigna a un vector *query* $q \in \mathbb{R}^{d_attn}$ y los *tokens* en el contexto son asignados a un vector *key* $k_t \in \mathbb{R}^{d_attn}$ y a un vector *value* $v_t \in \mathbb{R}^{d_attn}$. El producto punto $q^T k_t$ es interpretado como el grado en que el *token* $t \in V$ es importante para predecir el *token* q actual.

Alammar (2018) ofrece una explicación paso a paso del cálculo de la auto-atención utilizando la representación vectorial inicial del texto de entrada, formada por los *embeddings* de *token* y posición descritos anteriormente. El primer paso es la creación de los vectores *query*, *key* y *value* del vector de entrada. Para obtener estos vectores, es necesario multiplicar el vector de entrada por tres matrices cuyos coeficientes son aprendidos durante el proceso de entrenamiento de la red neuronal. Posteriormente, se calcula el puntaje de cada palabra del texto contra todas las palabras restantes, la puntuación determina cuánto enfoque colocar en otras partes de la oración de entrada a medida que codificamos una palabra en una posición determinada y se calcula mediante el producto punto $q^T k_t$ del vector *query* y el vector *key* de la palabra que estemos comparando. El siguiente paso es

dividir los puntajes entre 8, que es la raíz cuadrada de la dimensión de los vectores *key* utilizados en la implementación de Vaswani et al. (2017), sin embargo, pueden emplearse otros valores, y aplicar una función Softmax para la normalización de la puntuación y determinar la relevancia de cada palabra en su posición. El penúltimo paso consiste en multiplicar cada vector *value* por el puntaje softmax con el objetivo de mantener intacto el valor de las palabras relevantes y descartar aquellas palabras que sean irrelevantes. Finalmente, la última etapa en el calculo de la auto-atención consiste en realizar la suma de los vectores *value* ponderados.

La Figura 4.2 muestra un diagrama con todos los pasos para el calculo de la auto-atención para el texto de entrada compuesto por la frase de dos palabras Redes neuronales". Como se observa, se calcula el vector de atención para la primer palabra del texto.

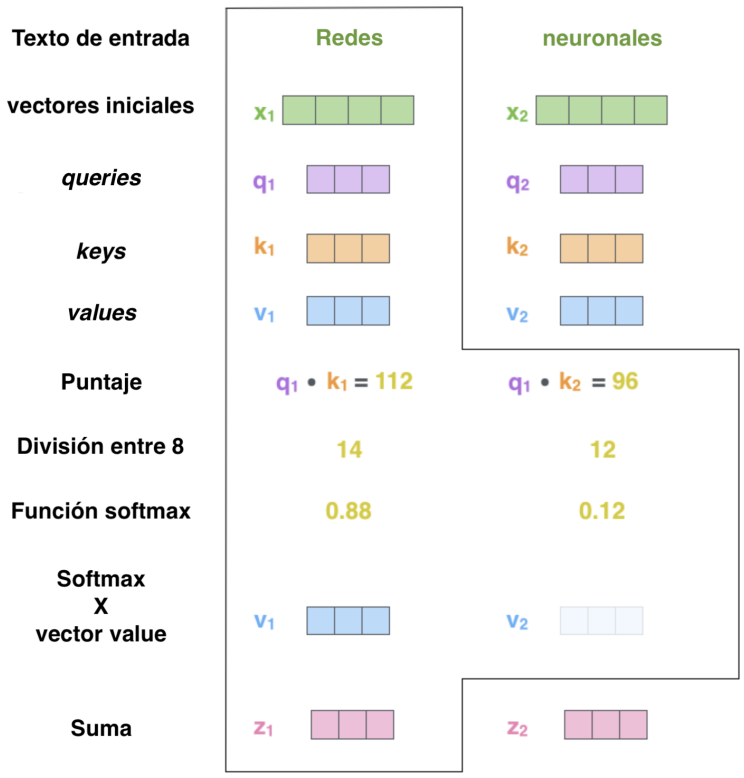


Figura 4.2: Diagrama de calculo de auto-atención utilizando la representación inicial del texto de entrada compuesto por las frase “Redes neuronales”. Imagen extraída de Alammar (2018).

El proceso de auto-atención descrito con anterioridad corresponde únicamente a una cabeza de atención, en la práctica, las redes *Transformer* emplean diversas cabezas de atención, cada una con pesos independientes que se aprenden durante el proceso de entrenamiento, y combinan sus salidas, técnica que se conoce como atención multi-cabeza.

4.2.4. Normalización

Finalmente, el último componente de la arquitectura *Transformer* es una normalización de capas que permite controlar de manera explicita la media y la varianza de las activaciones de la red neuronal. Además, de acuerdo con Ba et al. (2016) reduce el tiempo de entrenamiento de las redes neuronales y elimina la dependencia de los lotes utilizados en este proceso. Dentro de la normalización de capas todas las neuronas en una capa en particular tienen la misma distribución en todas las funciones para una entrada determinada. Por ejemplo, si cada entrada tiene d características, es un vector d-dimensional. Si hay elementos B en un lote, la normalización se realiza a lo largo del vector d-dimensional y no a lo largo del lote de tamaño B. La media y la varianza se calculan como se muestra en las ecuaciones 4.4 y 4.5 , respectivamente, donde H denota el número de neuronas de la capa. Por su parte, la ecuación 4.6 se utiliza para normalizar las entradas a las capas, donde ϵ es un parámetro que se añade cuando σ_l^2 es muy pequeño. Finalmente, la salida de la capa de la red neuronal se calcula mediante la ecuación 4.7 donde γ y β son parámetros que se ajustan durante el entrenamiento de la red.

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \tag{4.4}$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2} \tag{4.5}$$

$$\hat{x_i} = \frac{x_i - \mu^l}{\sqrt{\sigma_l^2 + \epsilon}} \tag{4.6}$$

$$y_i = \gamma\hat{x_i} + \beta \tag{4.7}$$

La Figura 4.3 muestra un diagrama de un bloque codificador de la arquitectura *Transformer* en el que se muestra cómo se aplican cada uno de los componentes descritos anteriormente. Como se observa, este corresponde al primer bloque de codificación, ya que tiene como entrada las representaciones vectoriales iniciales del texto conformada por los *embeddings* de *token* y de posición. De la misma forma, es importante señalar que cada etapa de normalización de capas cuenta con una conexión residual. Srivastava et al. (2015) señalan que este tipo de conexiones previenen el problema del desvanecimiento del gradiente, en el que las señales de la red tienden a cero durante la etapa de retro-propagación del error. Además, permiten a la red recordar el estado de las representaciones vectoriales de etapas previas, asegurando así que los *tokens* de entrada no sufran modificaciones.

Para el caso del modelo Sentence-CROBI se utilizan modelos de lenguaje pre-entrenados basados en la arquitectura *Transformer*, los cuales están compuestos por pilas de bloques como el de la Figura 4.3 y fueron pre-entrenados en grandes conjuntos de datos con el objetivo de aprender representaciones generales de los textos que puedan adaptarse con mayor facilidad a tareas específicas en las que los datos disponibles sean menores y así reducir el costo computacional necesario en términos de capacidad de almacenamiento y velocidad de procesamiento.

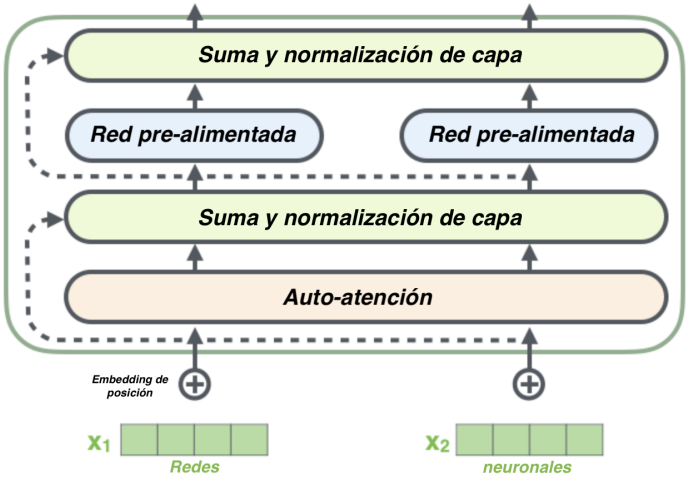


Figura 4.3: Diagrama del primer bloque codificador de la arquitectura *Transformer*. Imagen extraída de Alammar (2018).

4.3. Modelo Sentence-CROBI

En esta sección se presentan los componentes de la arquitectura Sentence-CROBI y su implementación. En primer lugar, el componente de bi-codificador de este enfoque se basa en la arquitectura Sentence-BERT (Reimers and Gurevych, 2019). Esta se compone de una modificación al modelo BERT (Devlin et al., 2019) mediante el uso de una red neuronal siamesa (Bromley et al., 1993); este tipo de redes se componen de dos redes de tipo *feed-forward* que comparten parámetros unidas a su salida. En el caso de Sentence-CROBI, a la salida de la componente siamesa se tienen vectores contextuales de palabras, a los que se les aplica una operación de *Pooling* para obtener una representación vectorial de cada texto, denotadas como u y v, respectivamente. En este trabajo se utiliza el modelo BERT-base (Devlin et al., 2019) como bi-codificador.

El segundo componente de la arquitectura propuesta es un codificador cruzado. Este codificador recibe la entrada conjunta de los pares de textos como se describe en la sección 4.1 y se toma como salida el último estado oculto del *token* especial de clasificación [CLS] como representación vectorial del par de textos. Para el desarrollo de esta tesis se prueban dos modelos distintos como codificadores cruzados: BERT-base (Devlin et al., 2019) y RoBERTa-large (Liu et al., 2019).

Después de obtener las representaciones individuales y conjunta del par de textos, se calcula la distancia euclidiana D_{uv} entre los vectores u y v. La distancia euclidiana se define como la longitud de un segmento de línea entre dos puntos en un espacio n-dimensional. Se define en la ecuación 4.8, donde: v_n y u_n representan la coordenada n-ésima en el espacio n-dimensional.

$$D_{uv} = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2} \tag{4.8}$$

Finalmente, las representaciones individuales u y v se concatenan al *token* especial de clasificación [CLS] de la representación conjunta y a la distancia euclideana D_{uv} para

formar la representación global del par de textos, la cual sirve como entrada para un bloque de clasificación cuya salida son las probabilidades de pertenencia a cada clase del problema, Paráfrasis y No Paráfrasis. La Figura 4.4 muestra el diagrama de alto nivel de la arquitectura Sentence-CROBI.

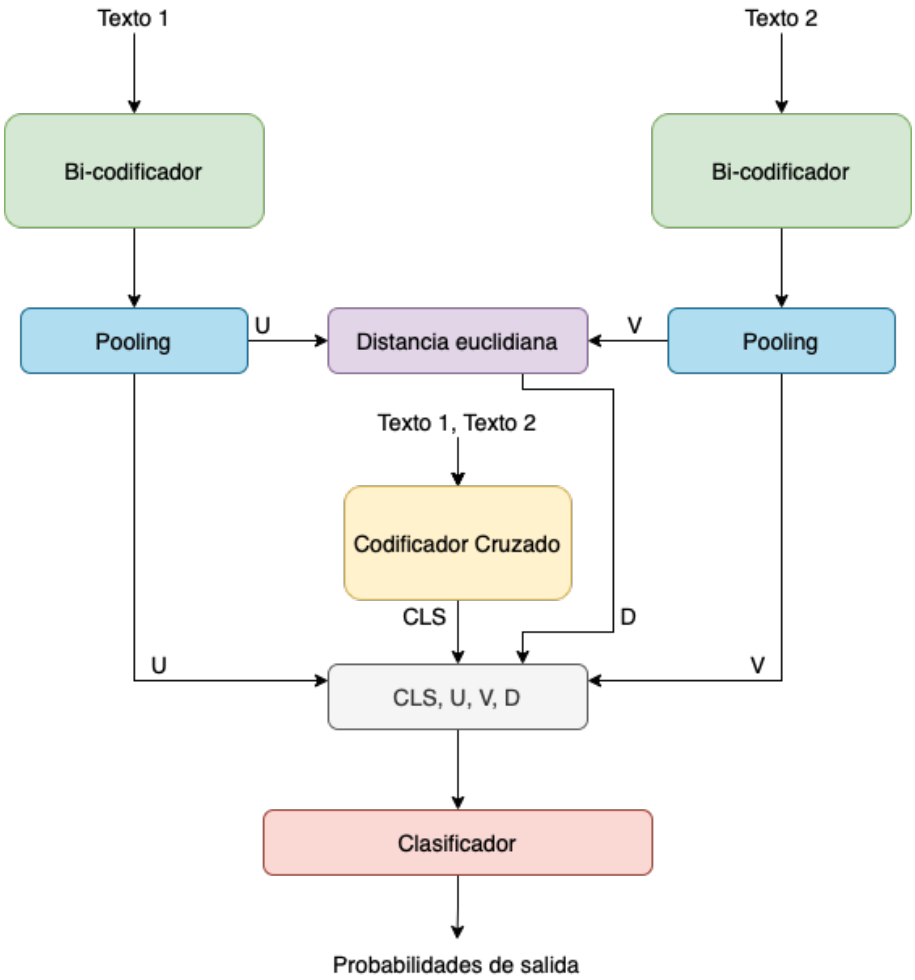


Figura 4.4: Diagrama de la arquitectura Sentence-CROBI. Imagen extraída de Ortiz-Barajas et al. (2022).

4.3.1. Vectores contextuales de palabras

La componente siamesa de la arquitectura Sentence-CROBI, compuesta por un bi-codificador produce a su salida vectores contextuales de palabras. Dado que los textos individuales son representados con una longitud fija de 35 *tokens* y se utiliza el modelo BERT-base (Devlin et al., 2019) en la componente siamesa, la dimensión de las matrices de los estados ocultos del modelo es de 35×768. Para determinar la mejor estrategia de obtención de los vectores contextuales de palabras, en este trabajo se probaron 3 estrategias diferentes para su obtención, como se muestra en la Figura 4.5; en esta se observan

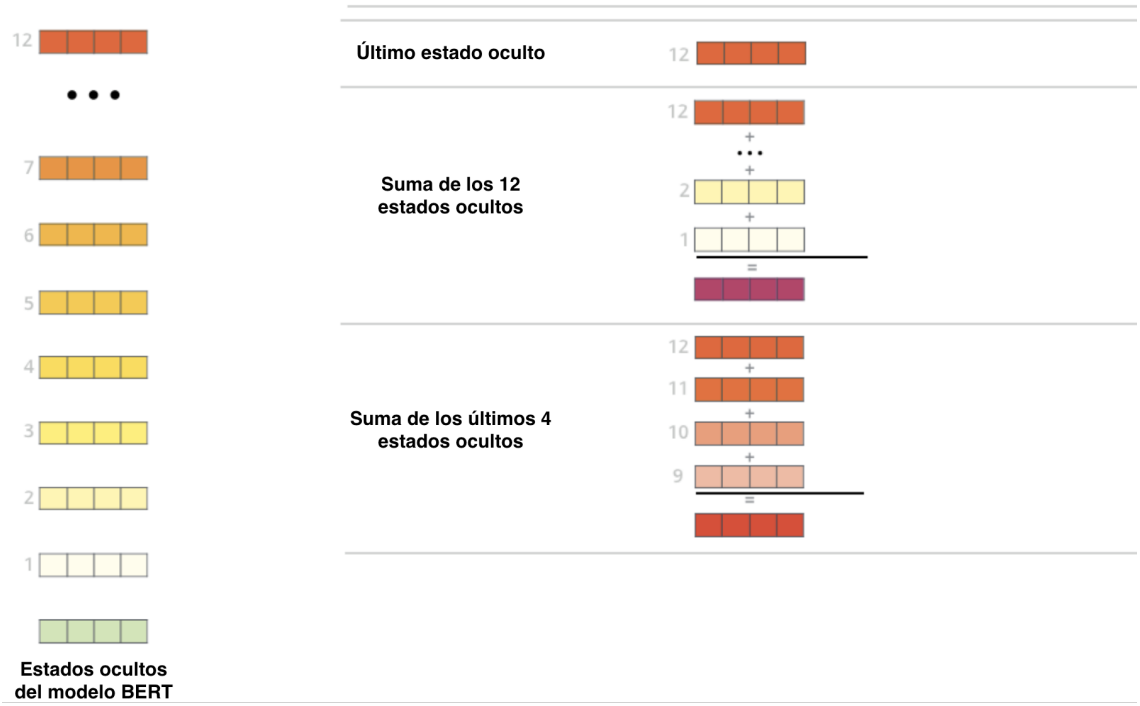


Figura 4.5: Diagrama de las diferentes estrategias utilizadas en la arquitectura Sentence-CROBI para obtener los vectores contextuales de palabras de la componente de codificador cruzado. Imagen extraída de McCormick (2019).

los 12 estados ocultos del modelo BERT-base (Devlin et al., 2019), correspondientes a los 12 bloques Transformer que componen esta versión. La primera estrategia consiste en tomar el último estado oculto del modelo, que es la salida del último bloque Transformer. La segunda estrategia empleada consiste en sumar todos los estados ocultos del modelo para obtener una representación final de cada palabra de los textos, y finalmente la tercera estrategia, que consiste en sumar únicamente los últimos 4 estados ocultos del modelo, correspondientes al segmento del bloque 9 al bloque 12 de los codificadores de este componente del modelo Sentence-CROBI.

4.3.2. Operaciones de *Pooling*

A las matrices de vectores contextuales de palabras que representan cada texto obtenidas mediante el bi-codificador de la arquitectura Sentence-CROBI es necesario aplicarles una función de *Pooling* con el objetivo de obtener una representación vectorial de cada texto. Dado que las matrices obtenidas son de dimensiones 35×768, a la salida de dicha función se trendrá un vector de dimensiones 1×768 por cada texto y se denotan como u y v, respectivamente.

De acuerdo con Goodfellow et al. (2016), una función de *Pooling* remplaza la salida de la red en una determinada etapa con un determinado con una estadística de resumen de

las salidas cercanas. Para el caso de este trabajo, se emplean dos tipos de funciones; *Mean Pooling*, en la que simplemente se calcula el promedio de todos los vectores de palabras, y *Max Pooling*, donde se selecciona el mayor valor de cada dimensión de todos los vectores contextuales de palabras. La Figura 4.6 muestra un diagrama con los dos tipos de *Pooling* utilizados en este trabajo. Como se observa en la Figura 4.6(a), para realizar la operación *Mean Pooling* se calcula el promedio de los valores a lo largo del eje y, es decir en la n-ésima dimensión de cada palabra que compone el texto. De forma análoga, la operación *Max Pooling* selecciona el valor máximo de la n-ésima dimensión de cada palabra que compone el texto.

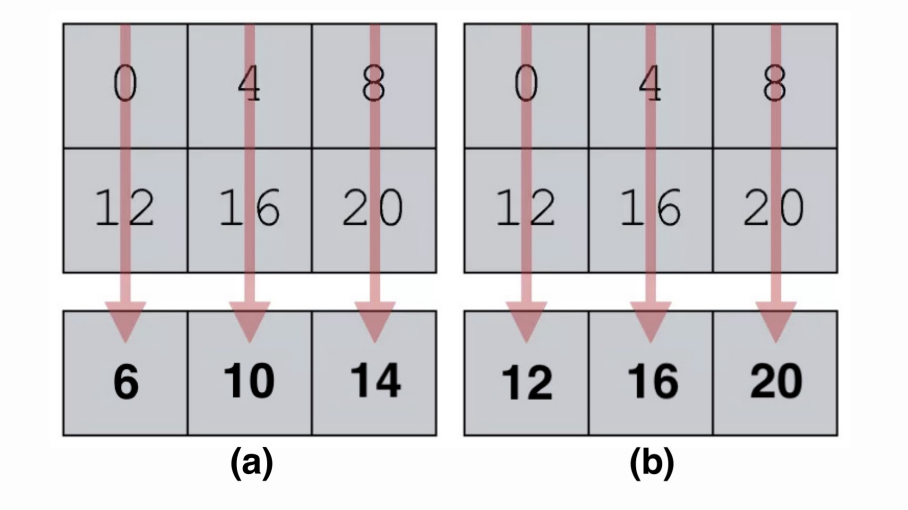


Figura 4.6: Operaciones *Mean Pooling* y *Max Pooling* utilizadas para obtener las representaciones individuales de los textos u y v.

4.3.3. Clasificador

El último componente de la arquitectura Sentence-CROBI es el bloque de clasificación. En este trabajo se probaron 3 bloques distintos que se explican a continuación. El primero de ellos es el más simple y consta de una capa completamente conectada con dos salidas con la función lineal como función de activación, que se define en la ecuación 4.9.

$$f(x) = x \tag{4.9}$$

El segundo bloque consiste en una red completamente conectada con 3 capas. Las primeras dos capas son idénticas y constan de 300 neuronas cada una, a sus salidas se aplica normalización por lote, que de acuerdo con Bjorck et al. (2018), es una técnica que permite mejorar el rendimiento del modelo y acelerar el entrenamiento ya que converge mas rápido a un valor de la función de pérdida, y se utiliza la función ReLU, que se define en la ecuación 4.10, como función de activación. La capa de salida tiene dos neuronas. El diagrama de este bloque de clasificación se muestra en la Figura 4.7(a).

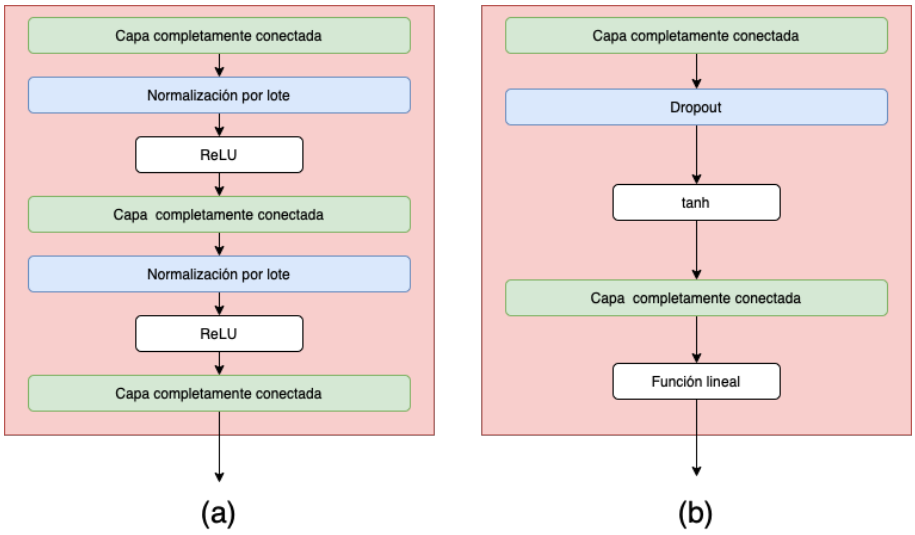


Figura 4.7: Segundo y tercer bloque de clasificación utilizados en la arquitectura Sentence-CROBI.

$$f(x) = \begin{cases} 0 & si \quad x < 0 \\ \\ x & si \quad x \geq 0 \end{cases} \tag{4.10}$$

Finalmente, el tercer bloque consiste en una red completamente conectada con dos capas. Primero, recibe la representación global del par de textos como entrada y se aplica una función de *dropout* con una probabilidad de 0.1.El *dropout* es una técnica de regularización para evitar el sobreajuste de la red; consiste en asignar a cero aleatoriamente algunos valores de su entrada. Luego, pasa a través de una capa completamente conectada de 1793 unidades con la función tangente hiperbólica, que se define en la ecuación 4.11, como función de activación. Finalmente, la capa de salida consta de 2 neuronas con una función lineal como función de activación. Su respectivo diagrama se observa en la Figura 4.7(b).

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.11}$$

4.3.4. Función de pérdida

Para llevar a cabo el entrenamiento de la arquitectura Sentence-CROBI se utilizaron dos funciones de pérdida. También llamadas funciones de error, estas permiten comparar la salida actual de la red con la salida deseada y evaluar el comportamiento de la red durante el periodo de entrenamiento.

La primer función de pérdida es la entropía cruzada. Esta función mide el rendimiento de un clasificador cuando la salida es una probabilidad entre 0 y 1. El valor de la función incrementa a medida que la probabilidad de salida de la red se aleja de la etiqueta real, de

forma que un modelo de clasificación perfecto tendrá una valor de 0 en la entropía cruzada. La ecuación 4.12 define la entropía cruzada, donde:

- y^i es la etiqueta real del ejemplo de entrenamiento.
- $\hat{y}^i$ es la probabilidad que predice el modelo.
- N es el tamaño del conjunto de evaluación.

$$EC = \sum_{i=1}^{N} y^i \log(\hat{y}^i) + (1 - y^i) \log(1 - \hat{y}^i) \tag{4.12}$$

Por otro lado, siguiendo el enfoque de Nicosia and Moschitti (2017), en este trabajo se utiliza una función de pérdida conjunta que además de minimizar la entropía cruzada, también utiliza una función de pérdida contrastiva (Hadsell et al., 2006), cuyo objetivo es comparar la distancia entre las representaciones de la red siamesa compuesta por el bi-codificador con la etiqueta real del ejemplo de entrenamiento. Esta función contrastiva se define en la ecuación 4.13, donde:

- y^i es la etiqueta real del ejemplo de entrenamiento.
- s_1 y s_2 son las representaciones individuales de los textos, es decir, u y v.
- $d(s_1^i, s_2^i)$ es la distancia euclidiana entre las representaciones vectoriales de los textos.
- El margen M es un hiperparámetro de entrenamiento que define un radio alrededor del espacio vectorial de s_1 y s_2, de forma que los pares disimilares contribuyen a la función de pérdida solamente si están dentro de ese margen.

$$L_c = \sum_{i=1}^{N} y^i d(s_1^i, s_2^i)^2 + (1 - y^i) max(M - d(s_1^i, s_2^i), 0)^2 \tag{4.13}$$

Por lo tanto, la función de pérdida conjunta utilizada se define en la ecuación 4.14, donde λ_c es un hiperparámetro y EC es la entropía cruzada definida en la ecuación 4.12.

$$L_{total} = \lambda_c L_c + EC \tag{4.14}$$

4.4. Ajuste fino

El ajuste fino de la arquitectura Sentence-CROBI para la tarea de identificación automática de paráfrasis se realiza mediante dos enfoques. El primero de ellos consiste en el enfoque original propuesto por los autores del modelo BERT (Devlin et al., 2019) y consiste en inicializar los parámetros tanto del codificador cruzado como del bi-codificador con los pesos aprendidos durante la etapa de pre-entrenamiento y entrenar el modelo completo por un número pequeño de épocas utilizando una tasa de aprendizaje pequeña. Sin embargo, de acuerdo con Chen et al. (2021), uno de los problemas con los modelos de

lenguaje pre-entrenados es que son propensos al sobreajuste cuando el conjunto de datos de la tarea objetivo es pequeño.

Debido al problema anterior, cuando se ajusta el modelo al corpus de paráfrasis de Microsoft (MRPC) que consta de 4,076 ejemplos de entrenamiento, se utiliza el enfoque propuesto por Phang et al. (2018), que consiste en un ajuste fino en dos fases mediante el uso de una tarea intermedia. Dicha tarea debe tener una mayor cantidad de datos etiquetados y estar relacionada con la tarea objetivo para poder incrementar la robustez y eficacia del modelo. Para esta tesis se utilizan los conjunto de datos *Multi-Genre Natural Language Inference* (MultiNLI) y *Quora Question Pairs*, descritos en el capítulo 3, como tarea intermedias.

4.5. Congelamiento de capas de los codificadores

De acuerdo con el trabajo de Lee et al. (2019), existe evidencia que sugiere que únicamente es necesario realizar el ajuste fino sobre algunas de las capas finales de un modelo de lenguaje pre-entrenado para obtener resultados de alta calidad en las tareas objetivo. Por este motivo, se sigue el enfoque propuesto por estos autores para llevar a cabo el proceso de ajuste fino de la arquitectura Sentence-CROBI, que consiste en congelar la capa de *embeddings*, así como los primeros n bloques *Transformer* encargados de la codificación de los pares de textos, tanto en el codificador cruzado como en el codificador.

4.6. Aprendizaje conjunto

Para mejorar el rendimiento de la arquitectura Sentence-CROBI en la tarea de identificación automática de paráfrasis, en esta tesis se utiliza la técnica *Bagging* (Breiman, 1996). Consiste en entrenar diferentes modelos de forma independiente y combinar cada conjunto de salidas generado para realizar una votación y obtener una predicción final.

De acuerdo con Goodfellow et al. (2016), para el caso de las redes neuronales artificiales, las diferencias en la inicialización aleatoria de algunos parámetros o en la generación de lotes de entrenamiento pueden provocar errores independientes en cada uno de los modelos utilizados en la técnica de aprendizaje conjunto, por lo que la predicción final con base en la votación sobre la salida de cada modelo puede tener un mejor rendimiento en comparación con cada modelo de manera individual.

Para utilizar está técnica de aprendizaje conjunto, se realiza lo siguiente:

1. Se lleva a cabo el ajuste fino en k instancias independientes del modelo Sentence-CROBI, cada una de ellas inicializada con una semilla aleatoria distinta.

2. Se calculan las probabilidades de salida para cada ejemplo del conjunto de prueba con cada una de las instancias independientes del modelo entrenadas. Como resultado de la evaluación, se obtienen k matrices de salida, cada una de estas contiene las probabilidades de pertenecer a cada clase de cada uno de los ejemplos y su dimensión es $N \times 2$, donde N representa el tamaño del conjunto de prueba utilizado mientras que 2 es el número de clases.

3. Finalmente, se calcula el promedio de probabilidad de las k predicciones y la clasificación se basa en la clase con la probabilidad más alta.

4.7. Detalles de entrenamiento

Siguiendo los enfoques de entrenamiento utilizados en los modelos BERT (Devlin et al., 2019) y RoBERTa (Liu et al., 2019), los cuales se utilizan como bi-codificador y codificador cruzado, respectivamente; en este trabajo se utilizan los siguientes hiperparámetros de entrenamiento.

- Un tamaño del lote en el rango $\{16, 32\}$.
- Una tasa de aprendizaje en el rango de $\{1 \times 10^{-5}, 2 \times 10^{-5}, 3 \times 10^{-5}\}$.
- Un optimizador Adam (Kingma and Ba, 2014) con una fracción de calentamiento de 0.06 y un decaimiento lineal a cero.
- 10 épocas de entrenamiento con un pseudo paro temprano con base en el rendimiento del modelo en los datos de validación.

Además, se utiliza la biblioteca Transformers (Wolf et al., 2020) de HuggingFace para la implentación de la arquitectura propuesta.

4.8. Evaluación

Finalmente, en esta sección se explican las métricas de evaluación utilizadas para estimar el rendimiento de la arquitectura Sentence-CROBI.

Hossin and Sulaiman (2015) señalan que uno de los aspectos más importantes para la obtención de un modelo óptimo en cualquier tarea de aprendizaje automático es la evaluación del mismo, por lo tanto se deben seleccionar las métricas adecuadas en función de la tarea a resolver. En el caso de las tareas de clasificación, en las que se asigna una etiqueta a cada instancia de entrada al modelo, las métricas de exactitud, precisión, exhaustividad son las más utilizadas. Dichas métricas se definen a través de una matriz de confusión, Sammut and Webb (2011) la definen como una matriz de dos dimensiones, indexada en una dimensión por la verdadera clase de la instancia y en la otra por la clase que le asigna el clasificador. La tabla 4.1 muestra un ejemplo de matriz de confusión para un problema de clasificación binaria. Se tienen 4 valores diferentes:

- Verdaderos positivos: Son los ejemplos con etiqueta positiva que el modelo clasificó correctamente.
- Falsos negativo: Son los ejemplos con etiqueta positiva que el modelo clasificó como negativos.
- Falsos positivos: Son los ejemplos con etiqueta negativa que el modelo clasificó como positivos.

- Verdaderos negativos: Son los ejemplos con etiqueta negativa que el modelo clasificó correctamente.

		Predicciones del modelo		
		Positivo	Negativo	Total
Valores reales	Positivo	VP	FN	$VP + FN$
	Negativo	FP	VN	$FP + VN$
	Total	$VP + FP$	$FN + VN$	N

Tabla 4.1: Matriz de confusión para un problema de clasificación binaria

4.8.1. Accuracy

La exactitud mide la proporción de predicciones correctas sobre el número total de instancias evaluadas, se denota como:

$$Accuracy = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.15}$$

4.8.2. Precision

La precisión mide la proporción de instancias positivas clasificadas correctamente entre el total de instancias positivas en la matriz de confusión. Se define como:

$$Precision = \frac{VP}{VP + FP} \tag{4.16}$$

4.8.3. Recall

La exhaustividad mide la cantidad de instancias positivas clasificadas correctamente entre la cantidad de instancias clasificadas correctamente. Se define como:

$$Recall = \frac{VP}{VP + FN} \tag{4.17}$$

4.8.4. F1-score

El valor de la medida F1 representa la media armónica entre la precisión y la exhaustividad. Se define como:

$$Medida - F1 = \frac{2 \cdot precision \cdot exhaustividad}{precision + exhaustividad} \tag{4.18}$$

4.9. Pruebas de significancia estadística

De acuerdo con Redman (2008), la significancia estadística permite determinar si un resultado probablemente se deba al azar o algún factor de interés. Cuando un descubrimiento es estadísticamente significativo, quiere decir que se puede estar seguro que es real, y no solamente buena o mala suerte con respecto a la elección de la muestra.

Berg-Kirkpatrick et al. (2012) explican el proceso de comparar un sistema nuevo A con un sistema base B. Asumiendo que existe un conjunto de prueba $x = x_1, x_2, \cdots, x_n$ en el que A supera a B por $\delta(x)$, una prueba de hipótesis permite descartar el caso en el que la victoria de A sobre B sea un evento improbable producto del azar mediante una hipótesis nula, denotada como H_0, en la que se asume que A no es mejor que B.

La prueba de hipótesis consiste en estimar esta probabilidad, denotada como $p(\delta(X) > \delta(x)|H_0)$, donde:

- X es una variable aleatoria sobre los posibles conjuntos de datos de tamaño n que pueden utilizarse.
- $\delta(x)$ es la ganancia real observada con base en la métrica de rendimiento utilizada.

Comúnmente, si la probabilidad $p(\delta(x) > \delta(x)|H_0) < 0.05$ es posible concluir que el valor observado de $\delta(x)$ es suficientemente improbable y por lo tanto, rechazar la hipótesis nula H_0, es decir, aceptar que A tiene un mejor que B y no solamente es producto del azar. La probabilidad $p(\delta(x) > \delta(x)|H_0)$ es denotada como $valor - p(x)$ o simplemente Valor-p. En muchos casos, el Valor-p no es fácil de calcular y se debe realizar una aproximación, la cual depende del tipo particular de prueba de hipótesis que se lleve a cabo.

Dror et al. (2018) proponen una guía para elegir el tipo de prueba de significancia estadística ideal con base en los escenarios más comunes en tareas de procesamiento del lenguaje natural. La Figura 4.8 muestra el árbol de decisión propuesto para la elección del tipo de prueba.

En primer lugar, es necesario conocer la distribución de la estadística del conjunto de datos de prueba $\delta(x)$ bajo la hipótesis nula H_0. En caso de que dicha distribución sea conocida, se debe usar una prueba paramétrica, las cuales permiten obtener resultados más poderosos (Fisher, 1936). Sin embargo, es un proceso muy complicado verificar que la estadística realmente tenga una cierta distribución. La prueba paramétrica más común es la prueba emparejada del estudiante-t, en la que se evalúa si las medias poblacionales de dos conjuntos de mediciones difieren entre sí y se basa en la suposición de que ambas muestras provienen de una distribución normal.

Por otro lado, en caso de no conocer la distribución, la mejor opción es utilizar una prueba no paramétrica, que no realiza ninguna suposición sobre la distribución. Las pruebas no paramétricas se pueden dividir en dos con base en su poder estadístico y su complejidad computacional. El primer grupo de pruebas no toma en cuenta los valores de las métricas de evaluación utilizadas para evaluar los sistemas A y B, sino que utiliza estadísticas de alto nivel y son denominadas pruebas libres de muestreo. El segundo grupo muestrea repetidamente del conjunto de prueba y utiliza las métricas de evaluación para calcular el Valor-p, por lo que su poder estadístico es mayor en comparación con las pruebas del primer grupo y se denominan pruebas basadas en muestreo, sin embargo el proceso de muestreo

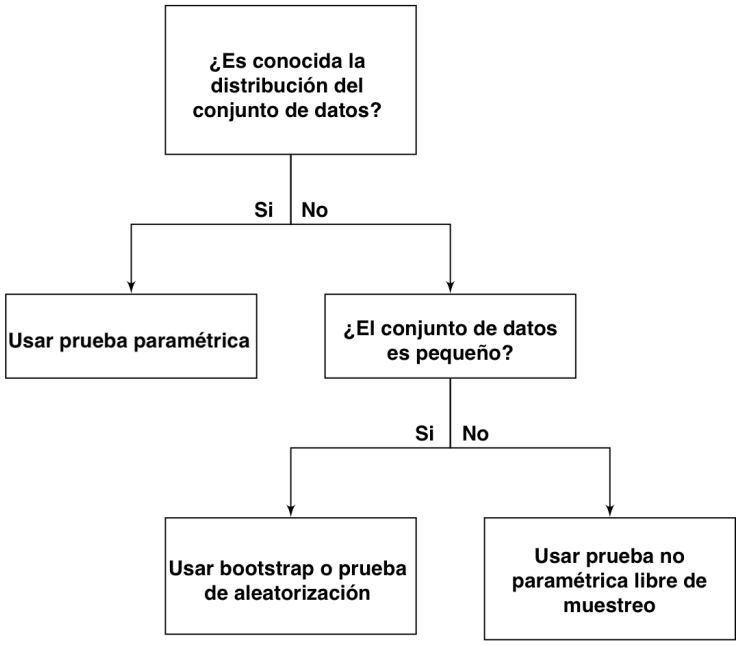


Figura 4.8: Árbol de decisión para seleccionar la prueba de significancia estadística adecuada. Imagen extraída de Dror et al. (2018).

repetido aumenta su complejidad computacional y no son ideales con conjuntos de datos grandes. Por lo tanto, el siguiente paso para seleccionar la prueba ideal, es necesario conocer el tamaño del conjunto de datos de prueba.

Las pruebas no paramétricas libres de muestreo más populares son cuatro. La primera es la prueba del signo, en la que se comprueba si las muestras de pares coincidentes se extraen de distribuciones con medianas iguales. $\delta(x)$ es el número de ejemplos para los cuales el modelo A es mejor que el modelo B y la hipótesis nula establece que dado un nuevo par de mediciones (por ejemplo, evaluaciones (a_i, b_i) de los dos modelos en un nuevo ejemplo de prueba), entonces a_i y b_i tienen la misma probabilidad de ser más grande que el otro. La segunda es la prueba de McNemar, que se utiliza para problemas de clasificación binaria y se utiliza una tabla de contingencia de 2×2 que compara las salidas de ambos modelos. La hipótesis nula para esta prueba establece que la probabilidad marginal para cada salida es la misma para ambos modelos. La tercera es la prueba Q de Cochran, que es una generalización de la prueba de McNemar para problemas multi-clase. Finalmente, la última prueba no parámetrica libre de muestreo es la prueba de los signos de Wilcoxon (Wilcoxon, 1946), cuya hipótesis nula es que las diferencias entre las muestras siguen una distribución simétrica alrededor de cero.

Las pruebas no paramétricas basadas en muestreo más utilizadas son la prueba de permutación de Pitman y la prueba de *Bootstrap*, que se basan en muestrear el conjunto de prueba repetidamente para calcular el Valor-p. Sin embargo, son computacionalmente muy costosos y son una opción poco viable aún con computadoras con alta capacidad de procesamiento y almacenamiento, cuando los conjuntos de datos son grandes.

Con base en la guía de Dror et al. (2018) se seleccionó la prueba de los signos de

Wilcoxon para realizar las pruebas de significancia estadística que comparan la arquitectura propuesta con los modelos del estado del arte.

4.9.1. Prueba de los signos de Wilcoxon

Demšar (2006) define la prueba de los signos de Wilcoxon (Wilcoxon, 1946) como la alternativa no paramétrica de la prueba t. Consiste en clasificar las diferencias en rendimiento de dos modelos para cada conjunto de datos, ignorando los signos y comparar las clasificaciones tanto para las diferencias positivas como negativas.

Sea d_i la diferencia entre los rendimientos de los dos modelos en la i-ésima salida de N conjuntos de datos. Las diferencias son clasificadas con base a los valores absolutos, y en caso de empates se asigna el promedio de la clasificación. La ecuación 4.20 denota la suma de los casos en los que el modelo A vence al modelo B, mientras que la ecuación 4.20 denota la suma de los puntajes del caso contrario, en los que el modelo B supera al modelo A. Como se observa en ambas ecuaciones, los casos en los que $d_i = 0$ se dividen equitativamente entre ambos casos.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{4.19}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{4.20}$$

Sea T el mínimo de ambas sumas $T = min(R^+, R^-)$. Una vez obtenido este dato, es posible calcular el Valor-p con base en la ecuación 4.21, donde n es el tamaño del conjunto de datos que se utiliza para la evaluación de los modelos A y B.

$$z = \frac{T - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{12}}} \tag{4.21}$$

Capítulo 5

Resultados

En este capítulo presentamos los resultados de la arquitectura Sentence-CROBI utilizando todas las configuraciones descritas en el capítulo 4. Los resultados presentados se dividen en dos grandes grupos.

En el primer conjunto, se muestran los resultados de los experimentos exploratorios realizados para conocer la efectividad de la arquitectura propuesta, para esta primera etapa, solamente se utilizó el corpus de paráfrasis de Microsoft (MRPC) para la evaluación del rendimiento y se comparan los resultados únicamente contra las componentes individuales de la arquitectura, es decir, el modelo BERT-base (Devlin et al., 2019) y el modelo Sentence-BERT-base (Reimers and Gurevych, 2019). Además, los resultados de este primer grupo de experimentos permitió descartar configuraciones en relación al bloque de clasificación empleado, la forma de representar los vectores contextuales de palabras, la operación de *Pooling* utilizada para obtener las representaciones individuales de cada uno de los textos, la función de pérdida con la que se lleva a cabo el ajuste fino y la cantidad de capas codificadores de cada componente que es necesario congelar para tener el rendimiento óptimo.

El segundo grupo de resultados corresponde a los experimentos con la versión final de la arquitectura Sentence-CROBI y su comparación con los enfoques del estado del arte, descritos en el capítulo 2. Para esta ronda de experimentos se consideran los tres conjuntos descritos en el capítulo 3: MRPC, QQP y PAWS-Wiki. Además, se consideran el rendimiento de todos los modelos obtenido con y sin el uso de técnicas aprendizaje conjunto. Finalmente, los resultados de este grupo son sometidos a pruebas de significancia estadística y se realiza un análisis cualitativo y cuantitativo de los errores del modelo.

Adicionalmente, es importante señalar que este trabajo ha sido parcialmente publicado en el número especial *Current Trends in Natural Language Processing (NLP) and Human Language Technology (HLT)* de la revista *Mathematics* (ISSN 2227-7390) bajo el titulo *Sentence-CROBI: A Simple Cross-Bi-Encoder-Based Neural Network Architecture for Paraphrase Identification* (Ortiz-Barajas et al., 2022).

5.1. Experimentos exploratorios

5.1.1. Primera ronda de experimentos exploratorios

En esta primera ronda de experimentos exploratorios, para la arquitectura Sentence-CROBI se utilizó el modelo BERT-base (Devlin et al., 2019) como codificador cruzado así como bi-codificador. Por su parte el clasificador corresponde al bloque más simple considerado en este trabajo, que consiste en una capa completamente conectada con la función lineal como función de activación. La configuraciones consideradas consisten en la forma de obtener los vectores contextuales de palabras, la función de pérdida utilizada y el uso del ajuste fino en dos fases como plantean Phang et al. (2018). Además se utilizó una tasa de aprendizaje α fija durante todo el entrenamiento.

La tabla 5.1 muestra los resultados de la primer ronda de experimentos exploratorios de la arquitectura Sentence-CROBI, correspondientes a una corrida simple del algoritmo de ajuste fino. Los resultados de los modelos Sentece-BERT (Reimers and Gurevych, 2019) y BERT-base (Devlin et al., 2019) corresponden al ajuste fino de estos modelos sobre el conjunto de datos MRPC utilizando sus implementaciones públicas disponibles. De forma general es posible observar que la arquitectura Sentence-CROBI supera el rendimiento del modelo Sentence-BERT (Reimers and Gurevych, 2019) en todas las configuraciones experimentales probadas. Por su parte, la arquitectura propuesta supera el rendimiento del modelo BERT-base (Devlin et al., 2019) en 4 configuraciones diferentes. La primera es cuando se utiliza el último estado oculto del bi-codificador como vectores contextuales de palabras y la entropía cruzada como función de pérdida. La segunda es cuando se aplica el algoritmo *Bagging* como técnica de aprendizaje conjunto tomando como modelos independientes cada instancia que obtiene los vectores contextuales de palabras de forma distinta. La tercera es empleando el ajuste fino en dos fases, tanto cuando se utiliza el conjunto QQP como MNLI como tareas intermedias, así como cuando se aplica el aprendizaje conjunto con las dos instancias anteriores.

Debido a que utilizar el último estado oculto obtiene los mejores resultados con la entropía cruzada y sin tareas intermedias, para las configuraciones siguientes únicamente se utilizó esta variante de los vectores contextuales de palabras. De forma análoga, a pesar de que al utilizar la pérdida conjunta se supera al modelo BERT-base, no supera el rendimiento al utilizar la entropía cruzada, por lo que para los experimentos de ajuste fino en dos fases, únicamente se utiliza la entropía cruzada como función de pérdida.

5.1.2. Segunda ronda de experimentos exploratorios

Para la segunda ronda de experimentos exploratorios de la arquitectura propuesta de nueva cuenta se utiliza el modelo BERT-base (Devlin et al., 2019) tanto en el componente de codificador cruzado, como en el bi-codificador. En esta etapa las diferentes configuraciones experimentales consisten en probar las diferentes funciones de *Pooling*, los diferentes bloques de clasificador propuestos y el congelamiento de las n primeras capas codificadoras del codificador cruzado y bi-codificador del modelo Sentence-CROBI. Tomando como precedente los resultados obtenidos en la primera ronda de experimentos exploratorios, para esta ronda únicamente se considera el último estado oculto del bi-codificador como

Modelo	Vectores Contextuales	Función de pérdida	Tarea Intermedia	Accuracy	F1-score
Sentence-BERT	-	EC	-	70.00	79.00
BERT-base	-	EC	-	83.00	88.00
Sentence-CROBI	UEO	EC	-	85.00	89.00
Sentence-CROBI	U4EO	EC	-	83.00	88.00
Sentence-CROBI	S12EO	EC	-	82.00	87.00
Sentence-CROBI	Conjunto	EC	-	84.00	89.00
Sentence-CROBI	UEO	PC	-	84.00	89.00
Sentence-CROBI	UEO	EC	QQP	85.00	89.00
Sentence-CROBI	UEO	EC	MNLI	85.00	89.00
Sentence-CROBI	UEO	EC	Conjunto	86.00	90.00

Tabla 5.1: Resultados de la arquitectura Sentence-CROBI y los modelos Sentence-BERT (Reimers and Gurevych, 2019) y BERT-base (Devlin et al., 2019) sobre el conjunto de datos MRPC. UEO corresponde al último estado oculto del modelo, U4EO corresponde a los últimos 4 estados ocultos del modelo y S12EO corresponde a la suma de los 12 estados ocultos del modelo. EC corresponde a la entropía cruzada y PC corresponde a la función de pérdida conjunta definida en la ecuación 4.14.

vectores contextuales de palabras y se utiliza la entropía cruzada como función de pérdida.

La tabla 5.2 muestra los resultados obtenidos por la arquitectura Sentence-CROBI en la segunda ronda de experimentos exploratorios. A diferencia de la primera ronda en la que se reporta el resultado de una corrida, para este caso se reporta la media de 5 corridas, utilizando una semilla aleatoria distinta para la inicialización de los modelos.

El primer grupo de resultados corresponde las variaciones entre las funciones de *Pooling* y el clasificador del modelo. Dado que no se realiza el congelamiento de ninguna capa ni se utilizan tareas intermedias para el ajuste fino, se utiliza un guión medio (-) para indicar esto. Además, el denotado como Bloque 2 en el clasificador del modelo, corresponde a un segundo bloque de clasificación descrito en la sección 4.3.3, que consiste en 3 capas completamente conectadas, con las primeras dos conformadas por 300 neuronas seguidas de normalización por lote y función ReLU como función de activación, y la tercera capa con dos neuronas y función lineal como función de activación. Mientras que el denotado como clasificador simple, corresponde al bloque de clasificación con una capa completamente conectada de dos neuronas. Es posible observar que el mejor rendimiento se obtiene con la función de *Mean Pooling* y el clasificador simple con 84.23 en *Accuracy* y 88.36 en *F1-score*. Además, se observa que el uso del bloque 2 de clasificación representa una disminución del rendimiento considerable, con una diferencia de 9.16 en *Accuracy* y de 6.85 en *F1-score* cuando se utiliza *Mean Pooling*, y de 5.82 y 4.53 cuando se utiliza *Max Pooling*, en *Accuracy* y *F1-score*, respectivamente.

Para el segundo grupo de resultados se toma como precedente los resultados del primer grupo, por lo que se utiliza la función *Mean Pooling* y el bloque de clasificación simple para todas las configuraciones posibles. En este grupo se sigue el enfoque de Lee et al. (2019) y se congela la capa de *embeddings* así como las primeras n capas del modelo tanto

del bi-codificador como del codificador cruzado. En este caso, cuando se indica un cero en las capas congeladas, significa que únicamente se ha congelado la capa de *embeddings*. Es posible observar que el mejor rendimiento de este grupo se obtiene precisamente en el caso anterior, cuando únicamente la capa de *embeddings* es congelada y sus parámetros no se actualizan durante el ajuste fino del modelo. Sin embargo, a pesar de que en términos de *F1-score* se tiene una mejora de 0.01 respecto a no congelar ninguna capa de los codificadores, hay una disminución de la eficacia de 0.07 en términos de *Accuracy*. Por otro lado, se observa una degradación del rendimiento conforme se aumenta el número de las primeras n capas congeladas que llega a su punto máximo cuando se congela la mitad de los bloques de codificación de la arquitectura propuesta, con unas métricas de 80.75 y 85.91, lo que representa una afectación de casi el 6 % y del 4 % en *Accuracy* y *F1-score*, respectivamente.

En el tercer grupo de resultados de nueva cuenta se toma el precedente del grupo anterior, por lo que se omite el congelamiento de las capas codificadoras del modelo y las configuraciones experimentales únicamente toman en cuenta diferentes conjuntos de datos para llevar a cabo el ajuste fino intermedio de dos fases descrito en la sección 4.4. De acuerdo con las métricas de rendimiento, es posible observar que el modelo que utilizan ajuste fino en dos fases con el conjunto MNLI como tarea intermedia supera al modelo con mejor rendimiento en los dos grupos anteriores. Existe una diferencia de 1.65 en términos de *Accuracy* y de 0.92 en función de *F1-score* con respecto al modelo que emplea *Mean Pooling* y un clasificador simple sin capas congeladas ni tareas intermedias y de 1.72 y 0.91 cuando se congela la capa de *embeddings*. Además, es importante resaltar que aunque los modelos que utilizan como tarea intermedia los corpus QQP y PAWS-Wiki tienen un rendimiento inferior, al utilizarlos en un enfoque de aprendizaje conjunto, obtienen el rendimiento más alto en esta segunda ronda de experimentos exploratorios con una media de *Accuracy* de 87.76 y de *F1-score* de 90.99, lo que indica de nueva cuenta que el algoritmo *Bagging* es una técnica efectiva de aprendizaje conjunto para elevar la eficacia en la tarea de identificación automática de paráfrasis.

Un aspecto interesante de los resultados de esta segunda ronda de experimentos exploratorios, es que el mejor rendimiento se alcanzó utilizado el bloque de clasificación simple, que es igual al propuesto por los autores del modelo BERT (Devlin et al., 2019) para tareas de clasificación. Por lo que para mantener la simpleza de la arquitectura Sentence-CROBI, en los experimentos posteriores se emplea el clasificador propuesto por los autores del clasificador cruzado.

Al concluir los experimentos exploratorios se seleccionaron las configuraciones experimentales que conforman la versión final de la arquitectura Sentence-CROBI para este trabajo, las cuales corresponden a lo siguiente:

- Utilizar el último estado oculto del bi-codificador como vectores contextuales de palabras.
- Emplear la entropía cruzada como función de pérdida durante el ajuste fino del modelo.
- Aplicar la función de *Mean Pooling* a los vectores contextuales de palabras para obtener la representación vectorial de cada texto.

Función de Pooling	Clasificador	Capas Congeladas	Tarea Intermedia	Accuracy	F1-score
Mean	Simple	-	-	84.23	88.36
Max	Simple	-	-	84.00	88.08
Mean	Bloque 2	-	-	75.07	81.51
Max	Bloque 2	-	-	78.41	83.83
Mean	Simple	0	-	84.16	88.37
Mean	Simple	1	-	84.07	88.14
Mean	Simple	2	-	83.49	87.93
Mean	Simple	3	-	83.11	87.59
Mean	Simple	4	-	82.74	87.41
Mean	Simple	5	-	82.12	86.89
Mean	Simple	6	-	80.75	85.91
Mean	Simple	-	QQP	83.95	88.23
Mean	Simple	-	MNLI	85.88	89.28
Mean	Simple	-	PAWS-Wiki	83.77	88.21
Mean	Simple	-	Conjunto	87.76	90.99

Tabla 5.2: Resultados de la arquitectura Sentence-CROBI sobre el conjunto MRPC variando la operación de Pooling y el clasificador del modelo. El denominado clasificador simple

- Ajustar los parámetros de todas las capas del modelo durante el ajuste fino.
- Utilizar el bloque de clasificación propuesto por los autores del clasificador cruzado.

5.2. Versión final de la arquitectura Sentence-CROBI

En esta sección se presentan los resultados de la versión final de la arquitectura Sentence-CROBI. En esta versión, para el codificador cruzado se utiliza el modelo RoBERTa-large (Liu et al., 2019), mientras que el bi-codificador es el modelo BERT-base (Devlin et al., 2019). El clasificador corresponde al bloque de clasificación 3 descrito en la sección 4.3.3 y consta de una red completamente conectada de dos capas, aplicando *dropout* a la entrada y utilizando las funciones de tangente hiperbólica y lineal como funciones de activación en la primera y segunda capa, respectivamente.

Esta versión de la arquitectura Sentence-CROBI se evalúa con los corpus de paráfrasis de Microsoft (MRPC), el conjunto de preguntas duplicadas de Quora (QQP) y el corpus PAWS-wiki. Además se compara contra los enfoques descritos en el capítulo 2.

5.2.1. Mejor rendimiento de cada modelo en los conjuntos MRPC y QQP

La tabla 5.3 muestra los resultados del estado del arte obtenidos de la clasificación pública ofrecida por el *GLUE Benchmark* y el rendimiento de la arquitectura Sentence-CROBI utilizando el corpus de paráfrasis de Microsoft (MRPC). Están ordenados de manera des-

cendente con base en la métrica *F1-score*. Los resultados del estado del arte son obtenidos a través de alguna técnica de aprendizaje conjunto, sin embargo, los autores no proveen las características detalladas de dicho proceso. Para la arquitectura propuesta, se reporta el resultado obtenido a través de la aplicación del algoritmo *Bagging* como técnica de aprendizaje conjunto; para ejecutar dicho algoritmo se utilizan 15 modelos independientes que corresponden a corridas utilizando diferentes semillas aleatorias. Cinco modelos corresponden a realizar el ajuste fino en dos fases, utilizando como tarea intermedia el corpus MNLI, otros cinco modelos son análogos a los primeros, pero utilizando el conjunto PAWS-Wiki como tarea intermedia en el ajuste fino de dos fases; y finalmente, los cinco modelos restantes corresponden al ajuste fino sobre el conjunto MRPC sin utilizar alguna tarea intermedia. Después de completar las 15 corridas, se calcula el promedio de todas las probabilidades de salida sobre el conjunto de prueba para obtener la predicción final.

La arquitectura Sentence-CROBI obtiene resultados competitivos en comparación con el estado del arte, ya que únicamente existe una diferencia de 1.23 en *Accuracy* y 0.75 en *F1-score* con respecto al modelo BORT (de Wynter and Perry, 2020), lider en la clasificación de este corpus. Con respecto a los enfoques ubicados en los lugares 2 a 6, hay en promedio una diferencia de 0.21 en *F1-score* y de 0.35 en *Accuracy*.

Model	**Accuracy**	**F1-score**
BORT	92.30	94.10
MT-DNN SMART	91.60	93.70
RoBERTa SMART	91.60	93.70
StructBERTRoBERTa	91.50	93.60
Funnel-Transformer	91.20	93.40
ALBERT	91.20	93.40
Sentence-CROBI	**91.07**	**93.35**
Ernie 2.0	87.40	90.20

Tabla 5.3: Resultados en el corpus de paráfrasis de Microsoft (MRPC) de la arquitectura Sentence-CROBI y los enfoques descritos en el estado del arte.

En la tabla 5.4 se pueden observar los resultados del estado del arte y la arquitectura Sentence-CROBI en el cojunto de preguntas duplicadas de Quora (QQP). A pesar de obtener resultados competitivos, existe una diferencia mayor entre el modelo propuesto y los enfoques del estado del arte, la cual es de 0.6 en *Accuracy* y 1.6 en *F1-score*. Para este conjunto de datos, los enfoques del estado del arte utilizan ajuste fino en una fase y aprendizaje conjunto de algún tipo. Para la arquitectura Sentence-CROBI, se utiliza aprendizaje conjunto mediante el algoritmo *Bagging*, empleando 5 modelos independientes obtenidos de entrenar sobre este conjunto de datos con semillas aleatorias diferentes. Además, es importante resaltar la dificultad que presenta el conjunto QQP debido a la diferencia entre las distribuciones de los conjuntos de entrenamiento y validación, y el conjunto de prueba.

La figura 5.1 presenta una gráfica de barras que muestra el mejor rendimiento de cada modelo en el conjunto de paráfrasis de Microsoft (MRPC) y el conjunto de preguntas duplicadas de Quora (QQP). El rendimiento de los enfoques del estado del arte es obtenido a través del servidor del *GLUE Benchmark*. Para el caso del conjunto MRPC, corresponden

Model	Accuracy	F1-score
Funnel-Transformer	90.70	75.40
StructBERTRoBERTa	90.70	74.40
ALBERT	90.50	74.20
RoBERTa SMART	90.01	74.00
MT-DNN SMART	90.20	73.90
Ernie 2.0	90.10	73.80
Sentence-CROBI	**90.10**	**73.80**
BORT	85.90	66.00

Tabla 5.4: Resultados en el conjunto de preguntas duplicadas de Quora (QQP) de la arquitectura Sentence-CROBI y los enfoques descritos en el estado del arte.

a modelos utilizando ajuste fino en dos fases y a alguna técnica de aprendizaje conjunto, mientras que para el conjunto QQP, solamente se utiliza ajuste fino en 1 fase y alguna técnica de aprendizaje conjunto. Es posible observar que todos los modelos alcanzan un *F1-score* mayor a 90 en el conjunto MRPC. Sin embargo, en el conjunto QQP únicamente el modelo BORT (de Wynter and Perry, 2020) obtiene un rendimiento menor a 70 en *F1-score*. La diferencia en el rendimiento de este modelo en los dos conjuntos de datos sugiere un inestabilidad en el proceso de ajuste fino del modelo debido al tamaño del modelo.

5.2.2. Media de rendimiento en los conjuntos PAWS-Wiki y MRPC

Para tener una comparación más completa, además de los experimentos anteriores en los que se reporta el mejor rendimiento de la arquitectura Sentence-CROBI y de los enfoques del estado del arte utilizando ajuste fino en dos fases y alguna técnica de aprendizaje conjunto, en esta sección presentamos los resultados obtenidos siguiendo un enfoque de modelo sencillo, en el que no se utiliza ninguna tarea intermedia para el ajuste fino ni se utiliza ninguna técnica de aprendizaje conjunto para mejorar el rendimiento. Para estos experimentos se utilizan únicamente los conjuntos MRPC y PAWS-Wiki, ya que son los únicos cuyas etiquetas del conjunto de prueba son públicas. De la misma forma, solo se reportan los resultados del estado del arte de aquellos enfoques que hacen pública su implementación. Finalmente, para estos experimentos se reporta la media de 5 corridas de cada modelo, utilizando semillas aleatorias diferentes.

La tabla 5.5 muestra los resultados de la arquitectura propuesta y el estado del arte en el conjunto PAWS-Wiki. En esta configuración experimental, el modelo Sentence-CROBI se coloca en el segundo lugar con un rendimiento de 94.80 en *Accuracy* y 94.21 en *F1-score* únicamente detrás del modelo RoBERTa (Liu et al., 2019) con ajuste fino mediante el algoritmo SMART (Jiang et al., 2020) cuyo rendimiento es de 94.93 y 94.34, lo que representa una diferencia de 0.13 tanto en términos de *Accuracy* como de *F1-score*.

La tabla 5.6 muestra los resultados en el conjunto MRPC con la configuración de modelo sencilla, es decir, sin tareas adicionales o técnicas de aprendizaje conjunto. El modelo Sentence-CROBI obtiene el tercer puesto de la clasificación con una diferencia de 0.02 y 0.01 en *Accuracy* y *F1-score* con respecto al modelo Ernie 2.0 (Sun et al., 2020), ubicado en el segundo puesto; y de 0.21 en *Accuracy* y de 0.08 en *F1-score* con el modelo

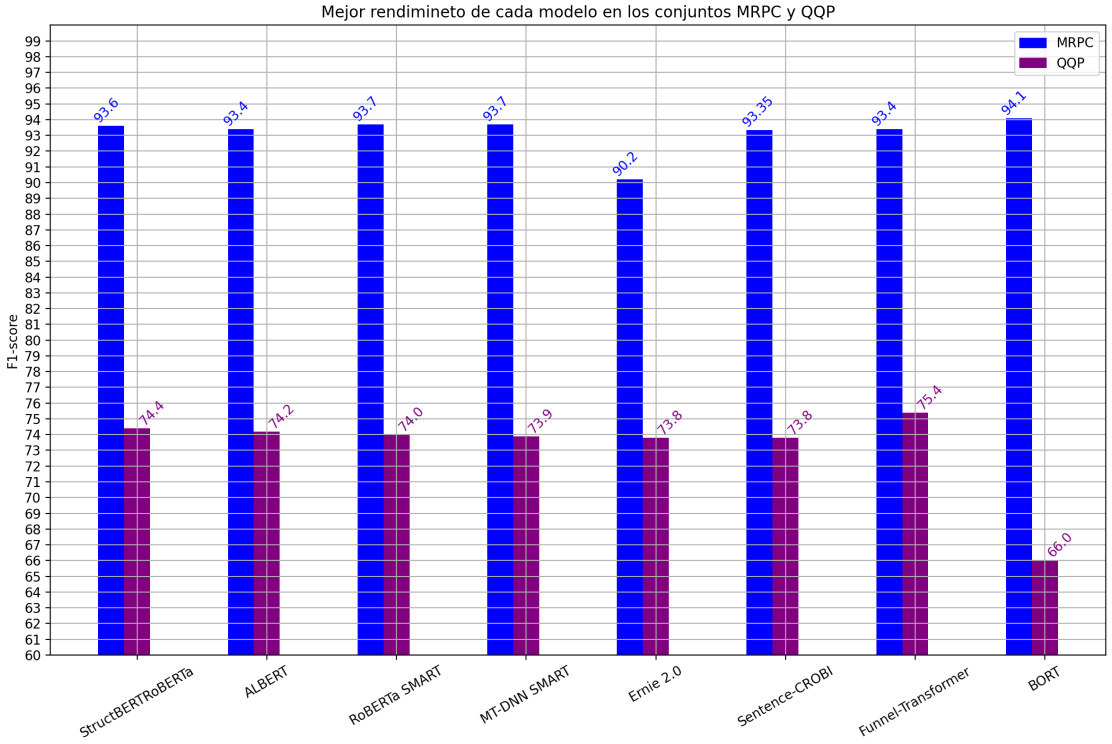


Figura 5.1: Métricas de mejor rendimiento de la arquitectura propuesta y el estado del arte en MRPC y QQP utilizando técnicas de ajuste fino intermedio y aprendizaje de conjunto.

Model	Accuracy	F1-score
RoBERTa SMART	94.93	94.34
Sentence-CROBI	**94.80**	**94.21**
DeBERTa	94.69	94.12
ALBERT	94.70	94.08
MT-DNN SMART	94.16	93.52
Ernie 2.0	93.86	93.18
StructBERT	93.13	92.41

Tabla 5.5: Resultados de la arquitectura Sentence-CROBI y el estado del arte en el conjunto de datos PAWS-Wiki

DeBERTa (He et al., 2020), líder de la clasificación. En ambos casos la diferencia es menor a 0.5 en ambas métricas, por lo que se muestra de nueva cuenta la competitividad de la arquitectura propuesta resaltando su facilidad de implementación, que no recae modificar la arquitectura Transformer (Vaswani et al., 2017), ni agregar tareas a la etapa de pre-entrenamiento como sí lo hacen los modelos en primero y segundo lugar, respectivamente.

Además, en comparación con los experimentos que reportan el rendimiento del mejor modelo con base en ajuste fino en dos fases y aprendizaje profundo, la arquitectura propuesta escala 4 posiciones en la tabla. Este hecho sugiere que la combinación de las representaciones del codificador cruzado y el bi-codificador proporciona una mejora considerable en la tarea de identificación automática de paráfrasis sin requerir esfuerzos adicionales en el proceso de ajuste fino o al emplear aprendizaje conjunto.

Model	Accuracy	F1-score
DeBERTa	89.30	91.96
Ernie 2.0	89.11	91.89
Sentence-CROBI	**89.09**	**91.88**
RoBERTa SMART	88.83	91.75
MT-DNN SMART	87.71	90.84
ALBERT	87.58	90.83
StructBERT	86.56	90.06

Tabla 5.6: Resultados de la arquitectura Sentence-CROBI y el estado del arte en el conjunto de datos MRPC

La figura 5.2 presenta una gráfica de barras que muestra el promedio de 5 corridas usando semillas aleatorias diferentes de la arquitectura Sentence-CROBI y los enfoques del estado del arte en los conjuntos MRPC y PAWS-Wiki. Todos los enfoques siguen una configuración de modelo simple sin tareas adicionales ni aprendizaje conjunto. Los modelos BORT (de Wynter and Perry, 2020) y Funnel-Transformer (Dai et al., 2020) no aparecen en esta gráfica ni en las tablas 5.6 y 5.5 debido a que sus implementaciones no están disponibles públicamente. Como se muestra, la arquitectura Sentence-CROBI está 0.56 por delante del promedio en *F1-score* en el conjunto MRPC, que es de 91.31. Además, 4 de los 7 modelos, incluida la arquitectura propuesta, tienen un rendimiento en *F1-score* mayor a 91 en el mismo conjunto de datos. De forma análoga, Sentence-CROBI supera al

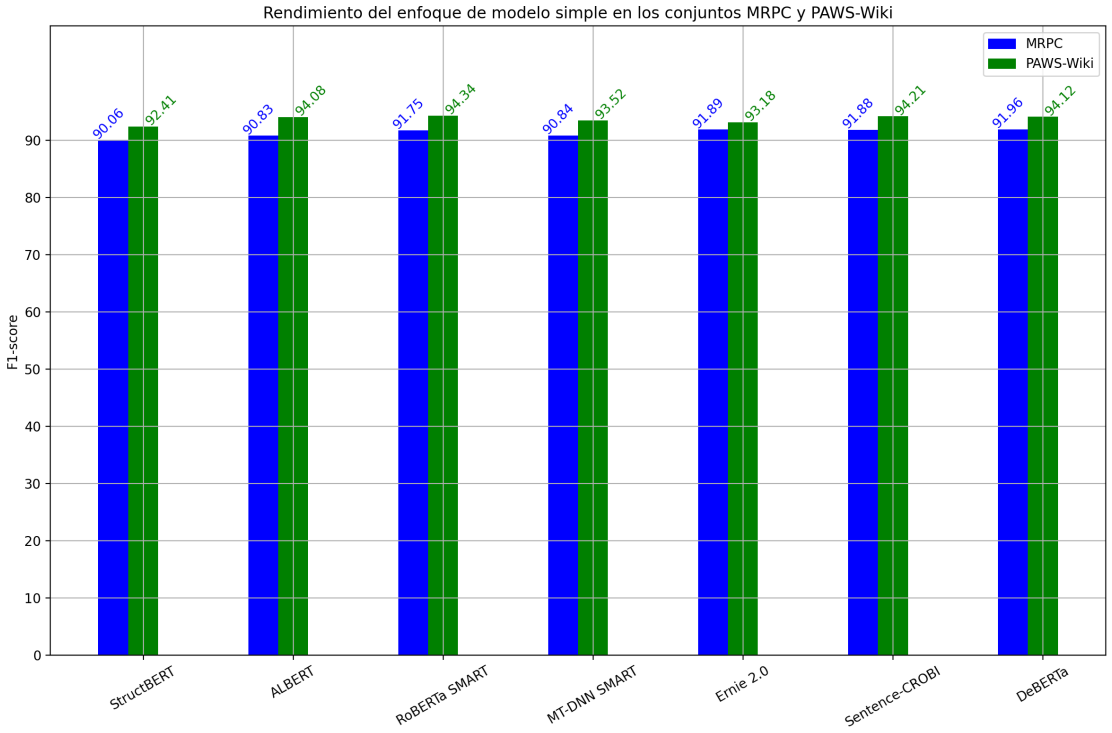


Figura 5.2: Rendimiento promedio de 5 corridas utilizando diferentes semillas aleatorias de la arquitectura Sentence-CROBI y el estado del arte sobre los conjuntos MRPC y PAWS-Wiki utilizando una configuración de modelo sencilla.

promedio en términos de *F1-score*, que es de 93.69 para el conjunto PAWS-Wiki por una diferencia de 0.51 y es además uno de los cuatro enfoques con un rendimiento mayor a 94 en dicho conjunto.

Esta comparación entre el estado del arte y la arquitectura propuesta siguiendo esta configuración de modelo sencilla en la que no se utilizan tareas adicionales durante el ajuste fino ni alguna técnica de aprendizaje conjunto para llevar a cabo la predicción final permite visualizar de una mejor forma la competitividad entre los diferentes enfoques debido a las diferencias mínimas que existen en los rendimientos. Además muestra la principal ventaja de la arquitectura Sentence-CROBI, la cual recae en su simpleza de implementación sin agregar cambios significativos a la arquitectura Transformer (Vaswani et al., 2017) ni al proceso de pre-entrenamiento de los modelos del lenguaje pre-entrenados.

5.3. Pruebas de significancia estadística

Adicional a la comparación mediante las métricas de rendimiento *Accuracy* y *F1-score*, se realizaron pruebas de significancia estadística entre la arquitectura Sentence-CROBI y los enfoques del estado del arte. Con base en la guía propuesta por Dror et al. (2018), se seleccionó la prueba no paramétrica de los signos de Wilcoxon (Wilcoxon, 1946) y se

utilizó la implementación en Python de la biblioteca SciPy (Virtanen et al., 2020). El umbral utilizado para aceptar o rechazar la hipótesis nula es $\alpha = 0.05$. Para llevar a cabo esta prueba se utilizan los resultados en los conjuntos MRPC y PAWS-Wiki siguiendo una configuración de modelo simple, es decir, sin tareas intermedias durante el ajuste fino ni algoritmos de aprendizaje conjunto.

La tabla 5.7 muestra los resultados de la prueba de los signos de Wilcoxon entre la arquitectura Sentence-CROBI y los enfoques del estado del arte. Como ser observa en las columnas MRPC Valor-p y PAWS-Wiki Valor-p, todos los Valores-p son mayores al umbral establecido α. Por lo tanto, no existe una diferencia significativa entre los modelos propuestos.

Modelo 1	**Modelo 2**	**MRPC Valor-p**	**PAWS-Wiki Valor-p**
Sentence-CROBI	ALBERT	0.0625	0.3125
	Ernie 2.0	0.8125	0.0625
	StructBERT	0.0625	0.0625
	RoBERTa SMART	0.3135	0.3125
	MT-DNN SMART	0.0625	0.0625

Tabla 5.7: Prueba de significancia estadística mediante la prueba de los signos de Wilcoxon entre la arquitectura Sentence-CROBI y el estado del arte. Se compara los Valores-p con el umbral $\alpha = 0.05$ para aceptar o rechazar la hipótesis nula.

Adicionalmente, se realizó la prueba de significancia estadística mediante la prueba de los signos de Wilcoxon comparando únicamente los enfoques del estado del arte entre sí. De la misma forma que con las pruebas con la arquitectura Sentence-CROBI, se utiliza un umbral $\alpha = 0.05$ para aceptar o rechazar la hipótesis nula, y las pruebas se realizaron utilizando los resultados obtenidos con los conjuntos MRPC y PAWS-Wiki sin tareas de ajuste fino intermedias ni algoritmos de ajuste fino intermedio. La tabla 5.8 muestra los valores-p de las comparaciones en ambos conjuntos de datos. Como ninguno de los valores son menores al umbral α no existe diferencia significativa entre los enfoques del estado del arte cuando resuelven la tarea de identificación automática de paráfrasis con los conjuntos MRPC y PAWS-Wiki.

De acuerdo con los resultados de las pruebas de los signos de Wilcoxon comparando la arquitectura Sentence-CROBI con el estado del arte de la tabla 5.7 y entre los enfoques del estado de la arte mostrados en la tabla 5.8 es posible concluir que no existe una diferencia estadísticamente significativa entre todos los enfoques comparados. Por lo tanto, la arquitectura propuesta tiene una ventaja con base en dos factores. El primero es la facilidad de implementación, que únicamente depende de utilizar dos modelos pre-entrenados, uno con un enfoque de codificador cruzado, y el otro como codificador y combinar su salidas para obtener una representación global, sin necesidad de modificaciones a la arquitectura Transformer (Vaswani et al., 2017) o la etapa de pre-entrenamiento. Además, esta arquitectura facilita la experimentación utilizando diferentes combinaciones de modelos. El segundo factor es el procedimiento de ajuste fino: Sentence-CROBI toma el esquema más sencillo mediante el uso de un número pequeño de épocas de entrenamiento, una tasa de aprendizaje baja y una función de pérdida estándar como la entropía cruzada para la tarea de identificación automática de paráfrasis.

Modelo 1	Modelo 2	MRPC p-Value	PAWS-Wiki p-Value
ALBERT	DeBERTa	0.0625	1.0
	Ernie 2.0	0.0625	0.0625
	StructBERT	0.1875	0.0625
	RoBERTa SMART	0.0625	0.0625
	MT-DNN SMART	1.0	0.0625
DeBERTa	Ernie 2.0	0.8125	0.0625
	StructBERT	0.0625	0.0625
	RoBERTa SMART	0.4375	0.125
	MT-DNN SMART	0.0625	0.0625
Ernie 2.0	StructBERT	0.0625	0.0625
	RoBERTa SMART	0.8125	0.0625
	MT-DNN SMART	0.0625	0.125
StructBERT	RoBERTa SMART	0.0625	0.0625
	MT-DNN SMART	0.0625	0.0625
RoBERTa SMART	MT-DNN SMART	0.0625	0.0625

Tabla 5.8: Resultados de las pruebas de significancia estadística mediante la prueba de los signos de Wilcoxon de los enfoques del estado del arte. Se utiliza un umbral $\alpha = 0.05$ para aceptar o rechazar la hipótesis nula.

5.4. Análisis del error

En esta sección se realiza un análisis de los errores cometidos por la arquitectura Sentence-CROBI en la tarea de identificación automática de paráfrasis en el corpus de paráfrasis de Microsoft. Dicho análisis se compone de una parte cuantitativa, en la que se ofrecen las estadísticas por cada clase del conjunto de datos; y una parte cualitativa, en la que se exploran las características de los ejemplos clasificador de manera errónea. Para ambos casos se utiliza los resultados obtenidos mediante el algoritmo *Bagging* de aprendizaje conjunto en el que se utilizan 15 modelos independientes inicializados con semillas aleatorias diferentes. Cinco modelos corresponden a aquellos en los que se utilizó el corpus MNLI como tarea intermedia durante el ajuste fino, en cinco más se utilizó el conjunto PAWS-Wiki como tarea intermedia, y los últimos cinco modelos considerados no emplean ninguna tarea intermedia.

5.4.1. Análisis cuantitativo

La figura 5.3 muestra la matriz de confusión obtenida por la arquitectura Sentence-CROBI mediante la configuración descrita anteriormente. Las predicciones finales fueron obtenidas al obtener el promedio de las probabilidades de salida de los 15 modelos considerados. Es posible observar que el modelo es capaz de identificar correctamente 1081 ejemplos de 1147 de la clase paráfrasis, que corresponde a al 94.24 % de las instancias de esta clase. Por otro lado, asigna de manera acertada la clase de No paráfrasis a 490 de las 578 instancias con esta etiqueta, porcentaje que representa el 84.77 % de eficacia en esta clase. Respecto a los ejemplos clasificados de forma incorrecta, hay 66 falsos negativos, es

decir, ejemplos de paráfrasis que la arquitectura no logró identificar. En el mismo sentido, hay 88 falsos positivos, que son ejemplos semánticamente equivalentes que el sistema no logró clasificar como paráfrasis.

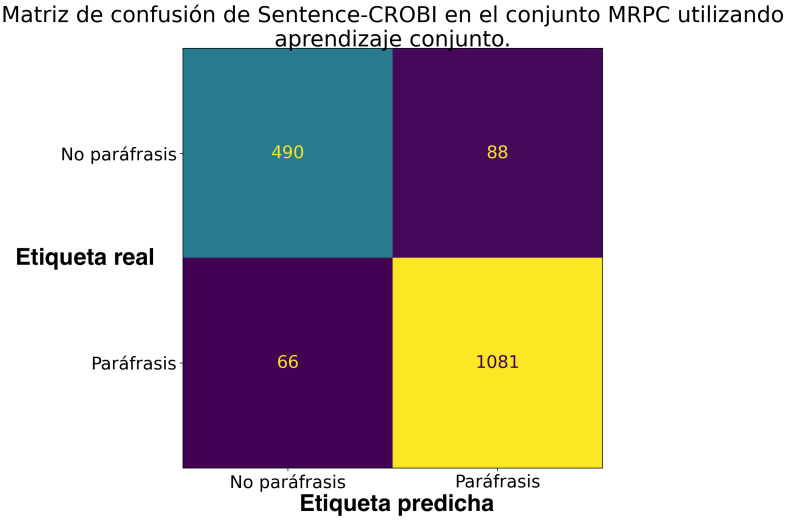


Figura 5.3: Matriz de confusión del modelo Sentence-CROBI en el corpus de paráfrasis de Microsoft mediante un enfoque de ajuste fino en dos fases y utilizando aprendizaje conjunto.

5.4.2. Análisis cualitativo

El análisis de error cualitativo consiste en identificar las características de los falsos positivos y falsos negativos clasificados por el modelo. Para llevar a cabo este proceso, se seleccionaron los primeros cinco ejemplos de cada conjunto y se discuten los aspectos interesantes de cada uno de ellos.

La tabla 5.9 muestra los ejemplos de falsos positivos. De forma general, es posible observar que todos los ejemplos comparten el sujeto. Por ejemplo, en el primer par de textos "Ballmer" es quien realiza la acción. En el segundo ejemplo, ambos hacen referencia a un sujeto femenino que acude al médico debido a que siente algún tipo de malestar. Finalmente, las diferencias entre los ejemplos tercero a quinto es el grado de detalle con el que se describen las acciones realizadas, sin embargo los sujetos son los mismos.

Por su parte, la tabla 5.10 muestra los ejemplos de predicciones falso negativas realizadas por el modelo propuesto. Sentence-CROBI tiene dificultad con los textos que tienen una superposición de palabras alta entre sí. Por ejemplo, en el primer ejemplo, el texto 1 habla sobre la posibilidad de que un hombre haya enfermado, mientras que el texto 2 habla del hecho de que un hombre se enfermó. Por otro lado, el tercer ejemplo es diferente por el número de cuerpos a los que se hace referencia en cada texto. Finalmente, en los ejemplos cuarto y quinto no logra identificar el rol semántico de los sujetos mencionados.

Texto 1	**Texto 2**
Ballmer has been vocal in the past warning that Linux is a threat to Microsoft.	In the memo, Ballmer reiterated the open-source threat to Microsoft.
"She first went to a specialist for initial tests last Monday, feeling tired and unwell".	"The star, who plays schoolgirl Nina Tucker in Neighbours, went to a specialist on 30 June feeling tired and unwell".
"Garner said the self-proclaimed mayor of Baghdad, Mohammed Mohsen al-Zubaidi, was released after two days in coalition custody".	Garner said self-proclaimed Baghdad mayor Mohammed Mohsen Zubaidi was released 48 h after his detention in late April.
Ït appears from our initial report that this was a textbook landing considering the circumstances", "Burke said".	"Said Mr. Burke: It was a textbook landing considering the circumstances"
"Powell recently changed the story, telling officers that Hoffa's body was buried at his former home, where the search was conducted Wednesday".	Powell changed the story earlier this year, telling officers that Hoffa's body was buried at his former home, where the aboveground pool now sits

Tabla 5.9: Ejemplos de falsos positivos del conjunto MRPC clasificados por el modelo Sentence-CROBI.

Texto 1	**Texto 2**
A Washington County man may have the countys first human case of West Nile virus, the health department said Friday.	The countys first and only human case of West Nile this year was confirmed by health officials on 8 September.
Snow's remark "has a psychological impact", said Hans Redeker, head of foreign-exchange strategy at BNP Paribas.	Snow's remark on the dollar's effects on exports "has a psychological impact", said Hans Redeker, head of foreign- exchange strategy at BNP Paribas.
Another body was pulled from the water on Thursday and two seen floating down the river could not be retrieved due to the strong currents, local reporters said.	Two more bodies were seen floating down the river on Thursday, but could not be retrieved due to the strong currents, local reporters said.
Amgen shares gained 93 cents, or 1.45 percent, to $65.05 in afternoon trading on Nasdaq.	Shares of Allergan were up 14 cents at $78.40 in late trading on the New York Stock Exchange.
In his speech, Cheney praised Barbour's accomplishments as chairman of the Republican National Committee	Cheney returned Barbour's favorable introduction by touting Barbour's work as chair of the Republican National Committee.

Tabla 5.10: Ejemplos falsos negativos predichos por la arquitectura Sentence-CROBI.

Capítulo 6

Conclusiones

En esta tesis se presenta el modelo Sentence-CROBI, una arquitectura de red neuronal simple basada en modelos de lenguaje pre-entrenados que emplea codificadores cruzados y bi-codificadores para obtener representaciones globales en tareas de pares de textos. El modelo combina de manera simple, a través de una concatenación y de una operación de distancia , los vectores conjuntos de un codificador cruzado y los vectores individuales de cada texto obtenidos a través de un bi-codificador. Por lo tanto, no depende de modificaciones a la arquitectura complejas, añadir nuevas tareas a la etapa de pre-entrenamiento, reducir el tamaño de un modelo o modificar el algoritmo de ajuste fino a tareas objetivo. Además, permite fácilmente combinar distintos modelos pre-entrenados, lo que ofrece la posibilidad de adaptarse a nuevas tareas y escenarios que involucren pares de enunciados.

La arquitectura propuesta obtiene resultados competitivos con el estado del arte en todos los conjuntos de datos utilizados. La diferencia más notable se presenta en el conjunto de preguntas duplicadas de Quora (QQP), donde el modelo se sitúa en el séptimo lugar de la clasificación y con una diferencia en términos de *F1-score* de 1.6 con el sistema de mejor rendimiento. Por otro lado, la menor diferencia se presenta en la evaluación con el conjunto PAWS-Wiki, donde el modelo RoBERTa (Liu et al., 2019) ajustado con el algoritmo SMART (Jiang et al., 2020) supera por 0.13 a la arquitectura propuesta, ubicándola en la segunda posición de la clasificación.

El rendimiento de Sentence-CROBI mejora considerablemente en comparación con el estado del arte cuando no se utiliza un esquema de ajuste fino en dos fases utilizando alguna tarea intermedia ni algoritmos de aprendizaje conjunto para realizar las predicciones finales. Estos resultados sugieren que la combinación de representaciones vectoriales basadas en codificadores cruzados y bi-codificadores pueden incrementar el resultado de un modelo sin utilizar técnicas adicionales. Además, no existe una diferencia estadísticamente significativa entre la arquitectura propuesta y el estado del arte. Este hecho representa una ventaja para este modelo, ya que su éxito no recae en agregar tareas de pre-entrenamiento, modificar la arquitectura Transformer (Vaswani et al., 2017), crear nuevos algoritmos de ajuste fino o aplicar alguna técnica de destilación del conocimiento. En ese mismo sentido, presenta una facilidad de implementación, ya que puede realizarse utilizando las herramientas existentes y una alta capacidad de adaptación a nuevas tareas. Dicha capacidad recae en llevar a cabo cambios mínimos, que consisten en cambiar la estrategia de combinación de las representaciones vectoriales, modificar el último bloque del modelo y emplear

una función de pérdida distinta. Esta configuración sigue el paradigma actual del campo del procesamiento del lenguaje natural en el que los modelos pre-entrenados son utilizados para resolver una amplia variedad de tareas sin necesidad de crear un modelo desde cero para cada una de estas.

6.1. Trabajo futuro

Este trabajo propone un nuevo eje de trabajo para enfoques basados en modelos de lenguaje pre-entrenados que consiste en la combinación de representaciones vectoriales de distintos tipos de arquitecturas para tareas de pares de textos. A partir de los resultados del modelo Sentence-CROBI en la tarea de identificación automática de paráfrasis, que consiste en un problema de clasificación binaria, es posible proponer algunas direcciones de trabajo futuro. La primera consiste en realizar una exploración exhaustiva de las diferentes combinaciones de codificador cruzado y bi-codificador utilizando modelos no basados en BERT. La segunda dirección de trabajo futuro consiste en proponer algoritmos para la selección óptima de hiperparámetros para la combinación de modelos pre-entrenados seleccionada. El tercer eje se centra en proponer tareas de pre-entrenamiento y algoritmos de ajuste fino para lograr una mejora estadísticamente significativa en comparación con el estado del arte y contribuir así a la creación de un sistema base para tareas a nivel de pares de enunciados. Finalmente, con relación al anterior, la última dirección de trabajo futuro a considerar es probar la arquitectura propuesta en tareas nuevas y en escenarios retadores.

Bibliografía

Alammar, J. (2018). The illustrated transformer. `https://jalammar.github.io/illustrated-transformer/`.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Berg-Kirkpatrick, T., Burkett, D., and Klein, D. (2012). An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Bhagat, R. and Hovy, E. (2013). What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese"time delay neural network. *Advances in neural information processing systems*, 6.

Chen, Y., Kou, X., Bai, J., and Tong, Y. (2021). Improving bert with self-supervised attention. *IEEE Access*, 9:144129–144139.

Dai, Z., Lai, G., Yang, Y., and Le, Q. (2020). Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in Neural Information Processing Systems*, 33:4271–4282.

de Wynter, A. (2020). An algorithm for learning smaller representations of models with scarce data. *arXiv preprint arXiv:2010.07990*.

de Wynter, A. and Perry, D. J. (2020). Optimal subarchitecture extraction for bert. *arXiv e-prints*, pages arXiv–2010.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Dror, R., Baumer, G., Shlomov, S., and Reichart, R. (2018). The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Fisher, R. A. (1936). Design of experiments. *British Medical Journal*, 1(3923):554.

Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Hamborg, F., Meuschke, N., Breitinger, C., and Gipp, B. (2017). news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

He, P., Liu, X., Gao, J., and Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Hossin, M. and Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1.

Humeau, S., Shuster, K., Lachaux, M.-A., and Weston, J. (2019). Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.

Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Zhao, T. (2020). SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Lee, J., Tang, R., and Lin, J. (2019). What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Markowitz, D. (2021). Transformers, explained: Understand the model behind gpt-3, bert, and t5. *daleonai.com*.

McCormick, C. (2019). Bert word embeddings tutorial. `https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/`.

Mota Montoya, M. A., Da Cunha, I., and Lopez-Escobedo, F. (2016). Un corpus de paráfrasis en español: metodología, elaboración y análisis. *RLA. Revista de lingüística teórica y aplicada*, 54(2):85–112.

Nicosia, M. and Moschitti, A. (2017). Accurate sentence matching with hybrid siamese networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 2235–2238, New York, NY, USA. Association for Computing Machinery.

Ortiz-Barajas, J.-G., Bel-Enguix, G., and Gómez-Adorno, H. (2022). Sentence-crobi: A simple cross-bi-encoder-based neural network architecture for paraphrase identification. *Mathematics*, 10(19).

Peng, Q., Weir, D., Weeds, J., and Chai, Y. (2022). Predicate-argument based bi-encoder for paraphrase identification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5579–5589.

Phang, J., Févry, T., and Bowman, S. R. (2018). Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Phuong, M. and Hutter, M. (2022). Formal algorithms for transformers.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Redman, T. C. (2008). *Data driven: profiting from your most important business asset*. Harvard Business Press.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sammut, C. and Webb, G. I. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., and Wang, H. (2020). Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Trinh, T. H. and Le, Q. V. (2018). A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Wang, W., Bi, B., Yan, M., Wu, C., Xia, J., Bao, Z., Peng, L., and Si, L. (2020). Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.

Wilcoxon, F. (1946). Individual comparisons of grouped data by ranking methods. *Journal of economic entomology*, 39(2):269–270.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhang, Y., Baldridge, J., and He, L. (2019). PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.