Universidad Zaragoza
1542

Trabajo Fin de Grado

Clasificación de relaciones léxico-semánticas con adapters para modelos de lenguaje

Relation classification with adapters for language models

Autor

Jorge Buil Lanau

Directores

Jorge Bernad Lusilla

Jorge Gracia del Río

Titulación

Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2023

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Jorge Buil Lanau ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Grado en Ingeniería Informática (Título del Trabajo)

Clasificación de relaciones léxico-semánticas con adapters para modelos de lenguaje

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 25 de agosto de 2023

Fdo: Jorge Buil Lanau

AGRADECIMIENTOS

Agradezco a mis directores, Jorge Bernad Lusilla y Jorge Gracia del Río, por su valiosa orientación y apoyo a lo largo de este trabajo, y por haberme introducido en este interesante campo de estudio.

También quiero mostrar mi agradecimiento a mi familia y amigos, por su constante apoyo y compañía en cada paso de este camino.

Clasificación de relaciones léxico-semánticas con *adapters* para modelos de lenguaje

RESUMEN

El objetivo de este trabajo es explorar el uso de *adapters* en la tarea de clasificación de relaciones léxico-semánticas, analizando sus ventajas y limitaciones. Los *adapters* son componentes adicionales que se conectan a un modelo de lenguaje preentrenado y permiten especializarse en tareas específicas sin perder el conocimiento previo del modelo.

En este trabajo, se ha realizado un estudio comparativo mediante el entrenamiento de modelos con y sin *adapters*, evaluando las métricas de rendimiento en diferentes conjuntos de datos. Los resultados sugieren que los *adapters* muestran una competencia comparable al enfoque convencional de ajuste fino y, en algunos casos, incluso presentan un rendimiento superior. Esto va acompañado de una reducción tanto en el tiempo de entrenamiento como en el espacio ocupado por el modelo durante el entrenamiento debido a la disminución de la cantidad de parámetros a ajustar.

En el marco de este estudio, se ha llevado a cabo un análisis exhaustivo de los resultados obtenidos, lo que ha permitido identificar patrones comunes de errores y áreas donde los modelos presentan un mejor desempeño. Además, se ha investigado el rendimiento de añadir *adapters* en capas específicas del modelo, observando mejor rendimiento en capas intermedias para algunos conjuntos de datos y en las primeras capas para otros. Sin embargo, se destaca la necesidad de realizar un análisis más profundo y exhaustivo en este aspecto.

También se ha identificado que la clasificación de sinonimia es particularmente desafiante. Se destaca la relevancia de la estrategia de *adapters* para lograr un equilibrio entre rendimiento y tiempo de entrenamiento, y se sugiere explorar la optimización de hiperparámetros, así como la evaluación de *adapters* en otros conjuntos de datos de clasificación de relaciones léxico-semánticas y probar diferentes plantillas de verbalización.

Índice

Capítulo 1

Introducción

1.1. Motivación

Modelos de lenguaje modernos como BERT [1] y RoBERTa [2] han demostrado ser altamente efectivos en diversas tareas de procesamiento del lenguaje natural, incluida la clasificación de relaciones léxico-semánticas.

Sin embargo, el enfoque convencional para abordar este problema implica el reentrenamiento de un modelo de lenguaje previamente entrenado con grandes cantidades de texto. Aunque este método ha llevado a la obtención de resultados sobresalientes, también tiene un efecto colateral significativo: se pierde parte del conocimiento previo del modelo durante el proceso de reentrenamiento.

La pérdida de conocimiento previo es una preocupación importante, ya que los modelos de lenguaje preentrenados contienen una amplia gama de información lingüística y conocimiento general adquiridos a partir de grandes corpus de texto. Este proceso de reentrenamiento podría impactar negativamente el rendimiento global del modelo, afectando su capacidad en diversas tareas para las cuales el modelo había sido competente previamente.

En este contexto, surge la motivación de explorar una alternativa al enfoque convencional de reentrenamiento de modelos de lenguaje. Se propone el uso de *adapters* [3], que son componentes adicionales que se conectan a un modelo de lenguaje. Estos *adapters* tienen como objetivo preservar el conocimiento previo del modelo mientras se especializan en una tarea específica, en nuestro caso la clasificación de relaciones léxico-semánticas.

Una ventaja clave de utilizar *adapters* es que requieren menos parámetros en comparación con el reentrenamiento completo de un modelo de lenguaje. Esto conlleva una reducción significativa en el tiempo de entrenamiento y en el espacio ocupado por el modelo durante el entrenamiento. Además, al preservar el conocimiento previo, los *adapters* pueden aprovechar la información lingüística y el conocimiento general

adquiridos por el modelo de lenguaje preentrenado, mejorando potencialmente el rendimiento en la clasificación de relaciones léxico-semánticas.

El objetivo principal de este trabajo es investigar si el uso de *adapters* puede ser una estrategia efectiva y eficiente para abordar la clasificación de relaciones léxico-semánticas sin perder el conocimiento previo del modelo de lenguaje. Se busca evaluar el rendimiento y las ventajas de esta aproximación en comparación con el enfoque de reentrenamiento convencional.

1.2. Objetivos y alcance

El objetivo principal de este estudio es investigar el uso de *adapters* en modelos de lenguaje preentrenados, como RoBERTa [2], para la clasificación de relaciones léxico-semánticas. Se evaluará el rendimiento y las ventajas de RoBERTa en la tarea de clasificación utilizando tres conjuntos de datos: EVALution [4], ROOT09 [5] y CogALex-V [6]. Se analizará cómo se comportan los modelos con y sin *adapters* en la clasificación de diferentes tipos de relaciones léxico-semánticas, como sinónimos, antónimos, hiperónimos y merónimos.

También se realizará un análisis detallado de los aciertos y errores de los modelos en la clasificación de relaciones léxico-semánticas, con el objetivo de comprender mejor las limitaciones y desafíos específicos de esta tarea. Este análisis permitirá identificar patrones comunes de errores y áreas en las que los modelos tienen un mejor desempeño, permitiendo una comprensión más precisa de las fortalezas y debilidades de los modelos en la clasificación de cada tipo de relación léxico-semántica.

Además, se realizarán experimentos para evaluar el rendimiento de modelos con *adapters* en diferentes capas del modelo. El objetivo es identificar las capas que brindan un mayor conocimiento y comprensión de las relaciones léxico-semánticas. Se investigará en qué niveles de representación del modelo se encuentran las características más relevantes para la tarea de clasificación, obteniendo información valiosa sobre las capas que poseen un mayor grado de conocimiento y contribuyen significativamente a la correcta identificación de las relaciones léxico-semánticas.

1.3. Metodología y herramientas

Este trabajo se estructura en cuatro partes principales. En primer lugar, se lleva a cabo un estudio y análisis de los modelos de lenguaje estilo BERT. A continuación, se realiza un análisis bibliográfico sobre las distintas estrategias de *adapters* utilizadas en modelos de lenguaje. Posteriormente, se realiza un análisis de los conjuntos de

datos disponibles en la comunidad para la clasificación de relaciones léxico-semánticas. Finalmente, se procede a la implementación y pruebas de los *adapters* para la clasificación de relaciones léxico-semánticas, incluyendo un análisis detallado de los resultados obtenidos.

Los datos, librerías y herramientas utilizadas son:

- EVALution, ROOT09 y CogALex-V: Conjuntos de datos de relaciones léxico-semánticas.
- R: Lenguaje y entorno para computación estadística y gráficos.
- Python: Lenguaje de programación del sistema.
- AdapterHub [7]: Marco de trabajo que simplifica la integración, el entrenamiento y el uso de *adapters* y otros métodos de ajuste fino eficientes para modelos de lenguaje basados en Transformer.
- `adapter-transformers`: Extensión de la biblioteca Transformers de Huggingface [8] que agrega componentes de *adapters* a los modelos de Transformer.
- Google Colab: Entorno colaborativo que ofrece al usuario un entorno de ejecución.
- Git y Github. Control de versiones y almacenamiento en la nube.
- Overleaf (LaTeX): Herramienta para la redacción de documentos.

El código fuente empleado en este trabajo, así como el material y los resultados generados, se encuentran disponibles en el siguiente repositorio: `https://github.com/jbuil/LexSem-Adapters` (ver estructura en el Anexo A).

1.4. Descripción del documento

El presente documento está organizado en 5 capítulos. En el capítulo 1, se encuentra la introducción, que proporciona una visión general del trabajo realizado. El capítulo 2 se enfoca en la presentación de conceptos básicos necesarios para comprender el contexto del resto del documento. En el capítulo 3, se aborda la selección de la arquitectura del sistema, los conjuntos de datos utilizados en el estudio, el proceso de entrenamiento de los modelos y la metodología utilizada para evaluar su rendimiento en la clasificación de relaciones. En el capítulo 4, se presentan los resultados obtenidos tras las evaluaciones, se realiza un análisis exhaustivo sobre dichos resultados y se investiga el rendimiento de modelos con *adapters* en distintas capas del modelo. Finalmente, en el capítulo 5, se exponen las conclusiones del trabajo y el posible trabajo futuro.

Capítulo 2

Marco teórico

En este capítulo se exponen los conceptos fundamentales necesarios para comprender el enfoque del trabajo. Se describen los modelos de lenguaje basados en Transformers, se presentan los *adapters* y se revisan varias arquitecturas propuestas en la literatura. Además, se aborda el tema de las relaciones léxico-semánticas, mencionando diferentes tipos de relaciones y diversos métodos de clasificación utilizados para categorizar estas relaciones.

2.1. Modelos de lenguaje

En esta sección, se ofrece una breve introducción a los modelos de lenguaje y se explora la arquitectura Transformer. Además, se examina el proceso de entrenamiento y clasificación de modelos estilo BERT, junto con una descripción del modelo RoBERTa.

2.1.1. Introducción a los modelos de lenguaje

Los modelos de lenguaje son un componente fundamental del procesamiento del lenguaje natural (PLN). En esencia, un modelo de lenguaje representa una entidad computacional que captura la estructura y gramática inherentes al lenguaje humano. Su objetivo principal es capturar la complejidad del lenguaje, permitiendo a las máquinas comprender y generar texto de manera más precisa y natural.

Este tipo de modelo se fundamenta en la capacidad de aprender las relaciones entre las palabras y las reglas que rigen su orden y contexto en un texto determinado. Esto se logra a través del entrenamiento con grandes cantidades de texto. A través del análisis estadístico de estos datos, el modelo aprende patrones y relaciones entre palabras, lo que le permite hacer predicciones sobre la probabilidad de ocurrencia de palabras o secuencias de palabras.

La importancia de los modelos de lenguaje en PLN radica en su capacidad para comprender el lenguaje humano en diferentes niveles, desde la sintaxis y la gramática

hasta el significado y la interpretación. Estos modelos son fundamentales para muchas aplicaciones de PLN, como la traducción automática, la generación de texto, el análisis de sentimientos y el resumen de texto, entre muchas otras.

2.1.2. Arquitectura Transformer

Existen diferentes enfoques para construir modelos de lenguaje. Algunos modelos se basan en n-gramas, donde se consideran secuencias de n palabras para predecir la siguiente palabra. Estos modelos son simples pero limitados en su capacidad para capturar las dependencias a largo plazo en el lenguaje. Por otro lado, las redes neuronales recurrentes (RNN) han sido ampliamente utilizadas en el procesamiento del lenguaje natural, ya que pueden modelar las dependencias secuenciales a través del tiempo. Sin embargo, las RNN tienen dificultades para capturar relaciones que se extienden a través de muchas palabras en el texto, y su entrenamiento puede ser más lento debido a la necesidad de tener en cuenta la información pasada para procesar la información actual. Es en este contexto donde las arquitecturas de Transformers han surgido como una alternativa poderosa y revolucionaria.

En 2017, Vaswani et al. [9] presentaron una arquitectura de codificador-decodificador (*encoder-decoder*) basada en capas de atención, que los autores llamaron Transformer (ver Figura 2.1). El nombre Transformer proviene de su habilidad de transformar una secuencia en otra.

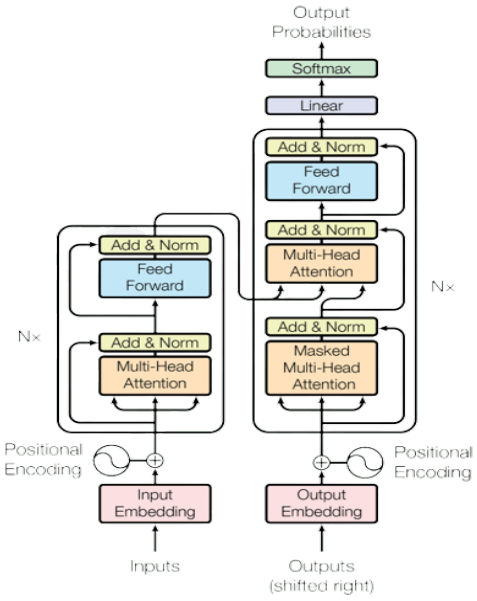


Figura 2.1: Arquitectura del modelo Transformer. (Fuente: Vaswani et al. [9])

Los componentes principales de los Transformers son la arquitectura codificador-decodificador, el mecanismo de atención y la autoatención.

La arquitectura Transformer toma una secuencia de datos de entrada, como texto, y la convierte en vectores que representan la semántica y la posición de cada palabra en la oración. Esta representación continua de la secuencia de entrada se conoce como *embedding*. El decodificador toma esta representación y genera la secuencia de salida paso a paso. En cada paso, considera la representación codificada y las salidas previas para predecir la siguiente palabra o elemento en la secuencia de salida. Durante la generación de la secuencia de salida, también se utiliza la autoatención para considerar las palabras generadas previamente. Tanto el codificador como el decodificador del Transformer constan de capas idénticas que incluyen mecanismos de atención y redes *feed-forward*.

La idea clave detrás del mecanismo de atención es que cada palabra o elemento en la secuencia de entrada puede contribuir a la representación de cada palabra o elemento en la secuencia de salida. En lugar de tratar todas las palabras o elementos de una secuencia de la misma manera, la atención asigna diferentes pesos a cada elemento en función de su importancia para la secuencia de salida.

La autoatención permite considerar las relaciones entre todas las palabras de una oración al mismo tiempo, asignando diferentes pesos a cada palabra según su relevancia para la palabra actual que se está procesando. Esto mejora significativamente la comprensión contextual y la captura de dependencias a lo largo del texto.

Ejemplos de modelos basados esta arquitectura incluyen BERT [1] y RoBERTa [2], los cuales se basan en la parte del codificador, y GPT [10], que se enfoca en el decodificador. A partir de este punto, nos enfocaremos en describir el entrenamiento y la clasificación de modelos estilo BERT.

2.1.3. Entrenamiento de modelos de lenguaje y clasificación

Los modelos de lenguaje basados en Transformers siguen un enfoque de aprendizaje semi-supervisado en su entrenamiento. En este proceso, la tokenización desempeña un papel crucial al dividir el texto en unidades más pequeñas llamadas "tokens", que pueden representar palabras, subpalabras o caracteres. Cada token tiene asociado un número único que corresponde a su representación en el vocabulario del modelo. Esta tokenización y procesamiento del texto son fundamentales, ya que los tokens procesados forman la base de entrada para el modelo.

Una vez preentrenados en un gran conjunto de datos no etiquetados, donde el modelo aprende a predecir palabras faltantes basándose en el contexto, adquieren un conocimiento general del lenguaje. En esta etapa, los modelos utilizan métodos como "*Masked Language Modeling*" (MLM), donde se enmascaran algunas palabras en una oración y el modelo debe predecir cuáles son esas palabras en función del contexto.

Cabe mencionar que el tamaño máximo de entrada que pueden manejar los modelos como BERT y RoBERTa es 512 tokens. Estos modelos utilizan un vocabulario de tokens que varía en tamaño, y en el caso de BERT, su vocabulario consta de 30522 tokens, mientras que RoBERTa utiliza un vocabulario aún más amplio, con 50265 tokens. Entre los tokens especiales utilizados en la tokenización se encuentran los por defecto llamados [CLS] y [SEP]. El token especial [CLS] se agrega al inicio de cada secuencia y se utiliza para obtener una representación de la secuencia completa que se utiliza para la clasificación de tareas. Por otro lado, el token especial [SEP] se utiliza para separar diferentes segmentos de texto en una sola entrada, como en el caso de secuencias de preguntas y respuestas.

Para adaptarse a tareas específicas, es necesario ajustar estos modelos preentrenados. El ajuste fino (*fine-tuning*) implica adaptar un modelo preentrenado a una tarea específica utilizando un conjunto más pequeño de datos etiquetados. Puede aplicarse en toda la red neuronal o en un subconjunto de sus capas, congelando las capas no ajustadas durante el proceso de retropropagación. El objetivo del ajuste fino es aprovechar el conocimiento general del modelo preentrenado y optimizar sus parámetros para mejorar el rendimiento en la tarea deseada. Esto ahorra tiempo y recursos en comparación con el entrenamiento desde cero.

Para que un modelo como BERT o RoBERTa puede llevar a cabo tareas de clasificación, se agrega una capa de clasificación. La cabeza de clasificación es una capa adicional que se conecta a la salida de la última capa del modelo.

Para llevar a cabo la clasificación, se emplea el vector asociado al token [CLS], el cual se conecta a la capa de clasificación, que consta de una capa densa y una capa de salida. Durante el ajuste fino, los parámetros de la cabeza de clasificación se optimizan para adaptarse a la tarea específica. La capa densa realiza transformaciones lineales y no lineales en este vector, permitiendo así capturar las características relevantes para la tarea de clasificación.

Finalmente, la capa de salida produce las probabilidades de clasificación. Si la tarea es de clasificación binaria, una sola neurona de salida utiliza la función de activación sigmoide para generar una probabilidad entre 0 y 1. Para clasificación multiclase, se utilizan múltiples neuronas de salida. Un ejemplo de función de activación adecuada en este caso es la *softmax*, que convierte las salidas en una distribución de probabilidades para todas las posibles categorías.

2.1.4. Modelo RoBERTa

El modelo RoBERTa (*Robustly Optimized BERT Approach*) es una mejora del modelo BERT (*Bidirectional Encoder Representations from Transformers*) propuesta

por Liu et al. [2] en 2019. Su objetivo principal fue optimizar y mejorar el rendimiento de BERT. Al igual que BERT, RoBERTa es un modelo de lenguaje basado en la arquitectura Transformer, centrándose en la parte del codificador o *encoder*.

RoBERTa fue entrenado utilizando el enfoque de MLM, similar a BERT. Durante esta fase, se enmascaran selectivamente palabras en las secuencias de entrada y el modelo se entrena para predecir las palabras enmascaradas utilizando el contexto. Sin embargo, a diferencia de BERT, RoBERTa empleó una variante en la que las palabras enmascaradas variaban entre diferentes instancias del mismo texto. Esto promovió una comprensión más profunda y generalizada del lenguaje, ya que el modelo no podía depender de una memorización excesiva. Además, RoBERTa utilizó tamaños de lote y secuencias más grandes, junto con un entrenamiento prolongado, abarcando una cantidad sustancialmente mayor de datos (160 GB, 10 veces más que BERT). Tal y como comentan los autores, se ha demostrado que RoBERTa supera a BERT y otros modelos de última generación en una variedad de tareas de procesamiento de lenguaje natural.

A continuación, se presentan dos variantes arquitectónicas del modelo RoBERTa. En la versión RoBERTa *base*, se emplean 12 capas Transformer, cada una con 12 cabezas de atención. El tamaño de la dimensión oculta es 768. Este modelo RoBERTa *base* tiene aproximadamente 125 millones de parámetros. Por otro lado, en la versión RoBERTa *large*, se emplean 24 capas Transformer, cada una con 16 cabezas de atención. El tamaño de la dimensión oculta es 1024. El modelo RoBERTa *large* contiene aproximadamente 355 millones de parámetros.

2.2. *Adapters*

En los últimos años, los modelos de lenguaje preentrenados han demostrado un gran éxito en una amplia variedad de tareas de procesamiento del lenguaje natural. Estos modelos, como BERT, RoBERTa y GPT, han sido entrenados en grandes cantidades de datos no etiquetados, lo que les permite capturar patrones lingüísticos y conocimiento general del lenguaje. Sin embargo, una característica de estos modelos es que, para tareas específicas, generalmente se requiere un ajuste fino o *fine-tuning* en un conjunto de datos anotado, lo cual puede ser costoso en términos de recursos computacionales y tiempo.

En un esfuerzo por abordar estas limitaciones, el concepto de *adapters* ha surgido como una técnica prometedora para mejorar la eficiencia y adaptabilidad de los modelos de lenguaje preentrenados. Los *adapters* son módulos ligeros que se insertan entre las capas del modelo preentrenado y permiten la adaptación para tareas específicas.

Durante el entrenamiento, solo se ajustan los parámetros del *adapter*, mientras que los parámetros del modelo preentrenado permanecen "congelados". Esto hace que los *adapters* sean mucho más eficientes, tanto en términos de tiempo como de almacenamiento, en comparación con el ajuste fino.

A fin de explorar y comprender mejor esta técnica, se van a analizar tres estudios destacados que han investigado y aplicado el uso de *adapters* en modelos de lenguaje natural, presentando diferentes arquitecturas y enfoques. Estos estudios incluyen el trabajo de Houlsby et al. [3], la propuesta de Pfeiffer et al. [11] y el enfoque de K-Adapter desarrollado por Microsoft Research [12]. Mediante el análisis y comparación de estos enfoques, se busca comprender cómo los *adapters* pueden mejorar la adaptabilidad y eficiencia de los modelos de lenguaje preentrenados en diversas tareas específicas.

2.2.1. Arquitecturas de *adapters*

El artículo de Houlsby et al. [3] plantea la ineficiencia de parámetros que surge al realizar el ajuste fino para varias tareas específicas de un modelo preentrenado. Comenta que si todos los parámetros de un modelo deben ser entrenados para cada tarea específica, el número de párametros a entrenar se dispara y por tanto surge una ineficiencia de parámetros.

Como alternativa, propone el uso de *adapters* para abordar este problema. Los resultados del estudio muestran que los *adapters* pueden lograr un rendimiento comparable al ajuste fino con un sustancialmente menor número de parámetros. En particular, los *adapters* alcanzan una puntuación media sobre el conjunto de evaluación GLUE [13] de 80.0, mientras que el ajuste fino logra 80.4. GLUE (*General Language Understanding Evaluation*) es un conjunto de datos de evaluación ampliamente utilizado que consiste en nueve tareas de clasificación de lenguaje y sirve como un indicador importante del rendimiento general del modelo en diversas tareas de comprensión del lenguaje natural.

La arquitectura de un módulo *adapter* y su integración en una capa de Transformer se muestra en la Figura 2.2.

Cada capa de un modelo basado en la arquitectura de Transformer está compuesta por dos módulos principales: un bloque de atención y un bloque *feed-forward*. Ambos módulos están seguidos por una conexión de salto (*skip-connection*). Como se muestra en la parte izquierda de la figura, Houlsby sugiere insertar una capa de *adapter* después de cada uno de estos bloques antes de la conexión de salto. El *adapter* se aplica directamente a la salida de cada bloque, después de la proyección de vuelta al tamaño de entrada, pero antes de agregar la conexión de salto. La salida del *adapter* luego se

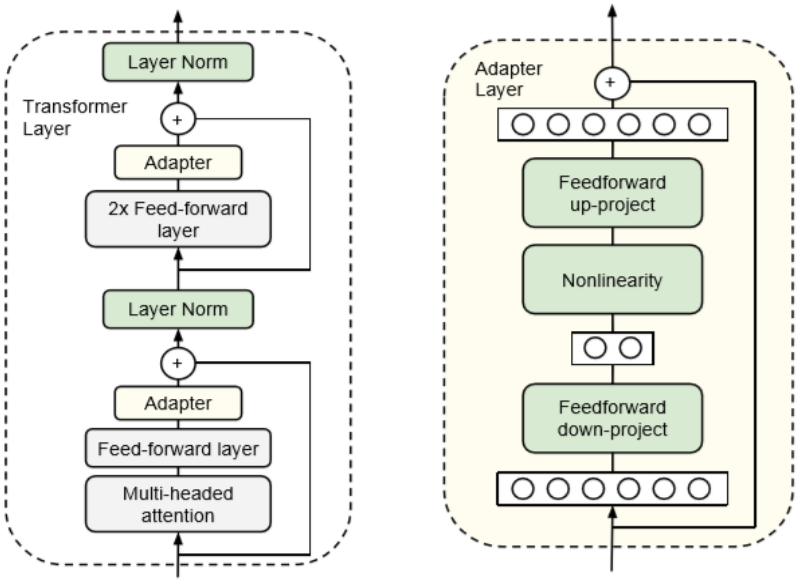


Figura 2.2: Arquitectura de un módulo *adapter* y su integración en una capa de Transformer. (Fuente: Houlsby et al. [3])

pasa directamente a la siguiente capa de normalización.

El módulo *adapter*, que se encuentra en la parte derecha de la figura, consta de una arquitectura de cuello de botella que contiene pocos parámetros en comparación con las capas de atención y *feed-forward* del modelo original. El *adapter* también incluye una conexión de salto.

La capa de *adapter* consiste en una proyección hacia abajo, una función no lineal y una proyección hacia arriba. La primera capa de proyección se utiliza para reducir la dimensión de las características de entrada a una dimensión más baja (llamada dimensión de cuello de botella) y la segunda para volver a aumentarla a la dimensión original. Un *adapter*, al mantener la dimensión de salida similar a la de su entrada, no produce ningún cambio en la estructura o parámetros del modelo original.

La cantidad de parámetros agregados por capa es $2md + d + m$, donde m es la dimensión a la que se proyectan las características, y d es la dimensión original. Al establecer $m \leq d$, se limita el número de parámetros agregados por tarea, lo que garantiza una adaptación eficiente.

La arquitectura de *adapters* propuesta por Pfeiffer et al. [11] se asemeja a la de Houlsby et al. en el sentido de que ambas utilizan *adapters* para adaptar los modelos de lenguaje preentrenados a tareas específicas. No obstante, difieren en el número de capas de *adapter* que colocan en cada capa del Transformer. Mientras que Pfeiffer et al. colocan una sola capa de *adapter* después del bloque *feed-forward*, Houlsby et al. utilizan *adapters* tanto después del bloque de atención como después del bloque *feed-forward* en cada capa.

Esta diferencia implica que la arquitectura de Pfeiffer emplea un menor número de *adapters*, concretamente la mitad, lo que reduce la cantidad total de parámetros añadidos por capa y resulta en una arquitectura más eficiente en términos de recursos computacionales y almacenamiento. Según el artículo [7], ambas arquitecturas muestran un rendimiento comparable en tareas específicas, a pesar de la diferencia en el número de *adapters* y parámetros utilizados.

En la Figura 2.3 se compara esquemáticamente una capa de un modelo Transformer sin *adapters* (Imagen A) con dos arquitecturas modificadas que incluyen *adapters*. La Imagen B muestra la arquitectura propuesta por Pfeiffer, mientras que la Imagen C representa la propuesta por Houlsby.

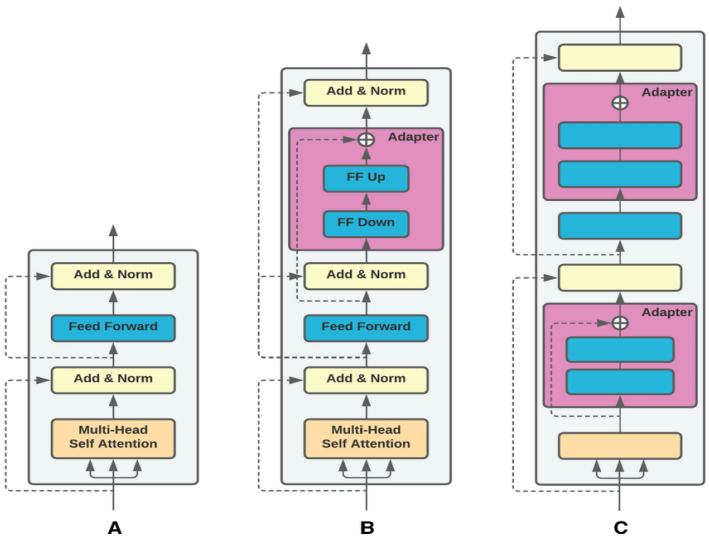


Figura 2.3: Comparativa de capa Transformer con *adapters*. (Fuente: Rathnayake et al. [14])

El artículo "K-Adapter: Infusing Knowledge into Pretrained Models with Adapters" [12] presenta una técnica desarrollada por Microsoft Research llamada K-Adapter, que aborda la limitación de la adaptabilidad de los modelos de lenguaje preentrenados a nuevas tareas y dominios. K-Adapter busca mejorar la adaptabilidad y el rendimiento al infundir conocimiento específico de la tarea en el modelo sin afectar su capacidad de tareas previamente aprendidas. A diferencia de las dos aproximaciones previamente analizadas, en las cuales se añaden *adapters* en cada capa del Transformer, en K-Adapter los *adapters* funcionan como *plug-ins* y se conectan externamente al modelo preentrenado (en el estudio, RoBERTa) sin modificar su estructura interna. Respecto a la arquitectura, cada capa de *adapter* contiene N capas de Transformer y dos capas de proyección, a las que se aplica una *skip-connection*. Los experimentos realizados en tres tareas dependientes del conocimiento (tipificación de entidades, respuesta a preguntas y clasificación de relaciones) demuestran que el rendimiento

de cada *adapter* logra una mejora significativa de manera individual y aún más cuando se combinan, capturando un conocimiento más rico en hechos y sentido común que el modelo preentrenado.

2.2.2. Ventajas del uso de *adapters*

Según comentan en el trabajo de Pfeiffer et al. [7], los *adapters* presentan ventajas significativas en comparación con el ajuste fino completo de los modelos de lenguaje. Los *adapters* son más eficientes en términos de parámetros, ya que solo actualizan una pequeña fracción de los parámetros del modelo, generalmente menos del 1 %. Además, son modulares, lo que permite extraer y compartir los parámetros actualizados de forma independiente del modelo base. Gracias a su tamaño de archivo reducido, se facilita su compartición y despliegue. Asimismo, los *adapters* aceleran el proceso de entrenamiento, requiriendo menos tiempo en comparación con el ajuste fino completo. Además, la capacidad de combinar múltiples *adapters* entrenados en diferentes tareas permite aprovechar su conocimiento combinado. A pesar de todas estas ventajas, los *adapters* logran ofrecer un rendimiento comparable al ajuste fino completo de los modelos de lenguaje.

2.3. Relaciones léxico-semánticas

En esta sección, se presentan diferentes tipos de relaciones léxico-semánticas y se exploran varios métodos de clasificación utilizados para categorizar estas relaciones.

2.3.1. Concepto y tipos

Las relaciones léxico-semánticas se refieren a las conexiones y asociaciones de significado entre las palabras, y cómo se relacionan entre sí en un contexto dado. Existen diferentes tipos de relaciones léxico-semánticas que se han investigado y analizado en el campo del procesamiento del lenguaje natural. Algunos de los tipos más comunes incluyen sinonimia, antonimia, hiperonimia, hiponimia y meronimia.

La sinonimia es una relación en la cual dos o más palabras tienen un significado similar o idéntico. Por ejemplo, las palabras “coche” y “automóvil” son sinónimas, ya que se refieren al mismo objeto. La sinonimia es importante en tareas como la recuperación de información, la expansión de consultas y la generación de lenguaje natural. Permite encontrar términos equivalentes y proporcionar variedad en la expresión lingüística.

La antonimia, por otro lado, se refiere a palabras que tienen significados opuestos. Por ejemplo, “bueno” y “malo” son antónimos, ya que representan dos extremos

opuestos de una cualidad. La antonimia es relevante en tareas como el análisis de sentimientos, donde se busca clasificar la polaridad de un texto.

La hiperonimia e hiponimia son relaciones que reflejan la jerarquía entre los términos. En la hiperonimia, una palabra representa una categoría o clase más amplia, mientras que en la hiponimia, una palabra es una instancia o subcategoría específica de esa clase. Por ejemplo, "animal" es un hiperónimo de "gato" y "perro", ya que abarca una categoría más amplia que incluye a estos animales específicos. La hiperonimia e hiponimia son importantes en tareas como la clasificación de textos y la generación de resúmenes, donde se necesita comprender las relaciones entre los conceptos y categorías.

La meronimia es una relación que se refiere a la relación parte-todo entre las palabras. En esta relación, una palabra representa una parte o componente de otra palabra, denominada ésta holónimo. Por ejemplo, "dedo" es un merónimo de "mano", ya que forma parte de la estructura de la mano. La meronimia es relevante en tareas que involucran el análisis y la representación de la estructura semántica de los objetos y conceptos.

La cohiponimia es una relación que se establece entre palabras que comparten un hiperónimo común. En esta relación, varias palabras se agrupan bajo un término más general que las engloba. Por ejemplo, "perro" y "gato" son cohipónimos, ya que ambos son ejemplos de animales domésticos. La cohiponimia resulta relevante en el análisis y la representación de la estructura semántica de las categorías y clasificaciones lingüísticas. Permite identificar y organizar palabras que comparten características similares pero que se diferencian en detalles específicos.

2.3.2. Métodos de clasificación

En el ámbito de la lingüística computacional, la clasificación de relaciones léxico-semánticas ha sido un tema de investigación importante para comprender las conexiones y significados entre las palabras.

Uno de los primeros enfoques utilizados en la clasificación de relaciones léxico-semánticas es el uso de patrones basados en reglas. Hearst propuso el uso de patrones de lenguaje específicos para identificar relaciones hiponímicas [15]. La detección de estos patrones permite inferir relaciones entre entidades nominales. Este enfoque se basa en la idea de que ciertas estructuras sintácticas y semánticas son indicativas de una relación hiponímica, donde una entidad es un subtipo o una instancia de otra.

Posteriormente, surgieron métodos basados en estadísticas que se centran en el análisis de la coocurrencia de palabras en un corpus. Estos enfoques utilizan técnicas estadísticas, como el cálculo de la Información Mutua Puntual (PMI), para descubrir

patrones de relación entre palabras. Estas medidas permiten identificar la relación entre palabras basándose en su frecuencia y asociación en el corpus [16].

Otro método importante son las representaciones distribucionales, como el *word embedding*. Estos métodos construyen representaciones vectoriales distribucionales de palabras utilizando técnicas como word2vec [17] o GloVe [18]. Estas representaciones capturan las relaciones semánticas basadas en la coocurrencia de palabras en el corpus. La similitud entre las representaciones vectoriales puede utilizarse para inferir relaciones léxico-semánticas.

Además, los recursos léxicos, como WordNet [19], juegan un papel crucial en la clasificación de relaciones léxico-semánticas, ya que proporcionan una estructura jerárquica y relaciones entre palabras. Estos recursos enriquecen el proceso de clasificación al ofrecer información semántica valiosa para comprender las relaciones entre términos.

Estos métodos representan solo una selección de diferentes enfoques utilizados en la clasificación de relaciones léxico-semánticas. A lo largo del tiempo, han surgido numerosas variaciones y mejoras de estos métodos, y también se han explorado enfoques híbridos que combinan múltiples técnicas para mejorar la precisión y cobertura en la clasificación de relaciones léxico-semánticas.

Capítulo 3

Diseño e implementación

En este capítulo, se detalla la arquitectura seleccionada para llevar a cabo el desarrollo de este trabajo. Además, se describen los conjuntos de datos utilizados en el proceso de entrenamiento y evaluación del modelo, junto con el enfoque empleado en dichas etapas, incluyendo métricas de evaluación relevantes.

3.1. Selección de la arquitectura

En el desarrollo de este trabajo se ha empleado AdapterHub [7], un marco de trabajo que simplifica la integración, entrenamiento y uso de *adapters* y otros métodos de ajuste fino eficientes para modelos de lenguaje basados en Transformers. Consiste en dos componentes principales: `adapter-transformers`, que es una extensión de la biblioteca Transformers de Huggingface [8] que agrega componentes de *adapters* a los modelos de Transformer, y "*The Hub*", que es un repositorio que recopila módulos de *adapters* preentrenados. AdapterHub ofrece configuraciones predefinidas para algunas arquitecturas de *adapters* propuestas en la literatura, incluyendo las de Pfeiffer et al. [11] y Houlsby et al. [3], así como la de He et al. [20], que coloca las capas de *adapter* en paralelo a las capas originales del Transformer. Esto brinda flexibilidad y facilidad para implementar diferentes arquitecturas de *adapters* en el sistema.

La arquitectura de *adapters* utilizada ha sido la propuesta por Pfeiffer. Esta elección se basa en las ventajas que proporciona esta arquitectura en términos de eficiencia de parámetros y tiempo de entrenamiento en comparación con otras alternativas, como la arquitectura propuesta por Houlsby. Al adoptar la arquitectura de Pfeiffer, se busca maximizar el rendimiento del modelo con la menor cantidad de recursos y tiempo de entrenamiento posible, además de que ambas arquitecturas ofrecen un rendimiento similar, tal y como se ha mencionado en el capítulo anterior.

El hiperparámetro más relevante a destacar en la configuración de los *adapters* es el factor de reducción. Este parámetro define la relación entre la dimensión oculta de

las capas del modelo y la dimensión del cuelo de botella (*bottleneck*) de los *adapters*. En la arquitectura propuesta por Pfeiffer, se ha utilizado un factor de reducción de 16.

Para la arquitectura de Pfeiffer, en cada capa de adapter se añaden dos capas de proyección. El número total de parámetros añadidos por capa, incluyendo los términos de sesgo (*bias*), es el siguiente:

- Para RoBERTa *large*: $2 \times 1024 \times 64 + 1024 + 64 = 132160$ parámetros (2 proyecciones, 1024 como dimensión oculta y 64 como dimensión de *bottleneck*).
- Para RoBERTa *base*: $2 \times 768 \times 48 + 768 + 48 = 74544$ parámetros (2 proyecciones, 768 como dimensión oculta y 48 como dimensión de *bottleneck*).

Considerando esta información, el número total de parámetros añadidos a RoBERTa *large*, a lo largo de sus 24 capas, es de 3171840 parámetros. Mientras tanto, RoBERTa *base*, con sus 12 capas, tiene un total de 894528 parámetros adicionales.

Además de esto, se entrena una capa final de clasificación. Esta consta de una capa lineal con $h \times h + h$ parámetros, seguida de una capa de salida con $h \times k + k$ parámetros. Aquí, h representa el tamaño de la dimensión oculta del modelo y k el número de grupos a clasificar.

3.2. Conjuntos de datos utilizados

En esta sección, se presentan los conjuntos de datos empleados en este trabajo para la clasificación de relaciones léxico-semánticas. Se han empleado tres conjuntos de datos distintos: EVALution [4], ROOT09 [5] y CogALex-V [6]. Cada conjunto de datos contiene diversas relaciones léxicas y semánticas que se utilizan para evaluar el rendimiento del modelo de lenguaje con la arquitectura de *adapters* propuesta.

Cada uno de los conjuntos de datos utilizados en este trabajo sigue una estructura uniforme que consiste en tres columnas: fuente, objetivo y relación. Para ilustrar esto, tomemos una instancia extraída del conjunto de datos EVALution, como se muestra en la Tabla 3.1:

Fuente	**Objetivo**	**Relación**
write	mark	synonym

Tabla 3.1: Ejemplo de estructura de los conjuntos de datos.

En este ejemplo, tenemos tres palabras: *write*, *mark* y *synonym*. La relación que se establece entre *write* y *mark* es que son sinónimos. Este formato de tripleta se mantiene constante en todos los conjuntos de datos.

A continuación, en la Tabla 3.2, se muestran las estadísticas de los conjuntos

de datos, detallando el número de pares para cada relación en las divisiones de entrenamiento, validación y test. Cabe mencionar que la relación 'desconocida' entre palabras se refiere a palabras que no están relacionadas semánticamente.

	EVALution	ROOT09	CogALex-V
Desconocida	-	4479/327/1566	2228/3059
Hiponimia	1327/94/459	2232/149/809	255/382
Cohiponimia	-	2222/162/816	-
Meronimia	699/41/231	-	163/224
Atributo	903/72/322	-	-
Antonimia	1095/90/415	-	241/360
Sinonimia	759/50/277	-	167/235
Posesión	377/25/142	-	-

Tabla 3.2: Estadísticas de los conjuntos de datos: Número de pares para cada relación en las divisiones de entrenamiento/validación/test.

3.2.1. EVALution

EVALution 1.0 [4] es un conjunto de datos diseñado para el entrenamiento y la evaluación de modelos de semántica distribucional (MSD). Esta versión consta de casi 7.5K tuplas que instancian varias relaciones semánticas entre pares de palabras, como hiperonimia, sinonimia, antonimia y meronimia. El conjunto de datos se enriquece con una gran cantidad de información adicional (como el dominio de la relación, la frecuencia de las palabras, la categoría gramatical, el campo semántico de las palabras, entre otros) que se puede utilizar para filtrar las parejas o realizar un análisis en profundidad de los resultados. Las tuplas se extrajeron de una combinación de ConceptNet 5.0 [21] y WordNet 4.0 [19], y posteriormente se filtraron a través de métodos automáticos y *crowdsourcing* para garantizar su calidad.

Con el objetivo de aumentar la homogeneidad de los datos y reducir su variabilidad, el conjunto de datos solo contiene pares de palabras cuyos términos (llamados *relata*) ocurren en más de una relación semántica. La información adicional se proporciona tanto para los términos como para las relaciones, y se basa en juicios humanos y datos del corpus. Es importante destacar que en este trabajo se ha realizado un análisis específico y centrado en los términos y la relación que existe entre ellos.

3.2.2. ROOT09

El conjunto de datos ROOT09 [5] se creó utilizando 9600 pares de palabras extraídos al azar de tres conjuntos de datos: EVALution [4], Lenci/Benotto [22] y BLESS [23]. Los pares de palabras se distribuyen de manera equitativa en tres clases: hiperónimos,

cohipónimos y palabras aleatorias, y abarcan varios tipos de palabras como adjetivos, sustantivos y verbos.

En cuanto a la composición del conjunto de datos, la clase de hipónimos contiene 2447 pares de sustantivos, 458 pares de verbos y 295 pares de adjetivos. La clase de cohipónimos tiene solo 3200 pares de sustantivos, los cuales fueron obtenidos exclusivamente de BLESS, ya que esta relación no existe en los otros dos conjuntos de datos. La clase de palabras aleatorias contiene 1100 pares de sustantivos, 1050 pares de verbos y 1050 pares aleatorios.

El conjunto de datos completo consta de 4263 términos, incluyendo 2380 sustantivos, 958 verbos y 927 adjetivos. En promedio, cada término aparece 4.5 veces. Si consideramos solo la primera palabra de los pares, hay 1265 términos diferentes (987 sustantivos, 186 verbos y 92 adjetivos). Por otro lado, si consideramos solo la segunda palabra, hay 3665 términos diferentes (1945 sustantivos, 860 verbos y 862 adjetivos).

Además, en un análisis adicional, se amplió este conjunto de datos intercambiando al azar los 3200 pares de hiperonimia (por ejemplo, de "*car HYPER vehicle*" a "*car RANDOM mammal*"). Este proceso se realizó con el propósito de evaluar la capacidad del sistema para clasificar correctamente estos pares como palabras aleatorias.

3.2.3. CogALex-V

El conjunto de datos CogALex-V [6] se creó como parte de la 5ª edición del taller *Cognitive Aspects of the Lexicon* (CogALex-V) con el objetivo de proporcionar una referencia común para probar los métodos actuales basados en corpus para la identificación de relaciones léxico-semánticas (sinonimia, antonimia, hiperonimia, meronimia) y comprender mejor sus fortalezas y debilidades respectivas. El conjunto de datos utilizado en el desafío se extrajo de EVALution 1.0 [4], el cual consta de pares de palabras que tienen las relaciones mencionadas anteriormente, así como elementos de control no relacionados semánticamente (aleatorios). La tarea a realizar se descompuso en dos subtareas: en primer lugar, detectar para cada pareja de palabras si estaba relacionada semánticamente o no; en segundo lugar, para cada pareja de palabras, determinar la relación semántica que existía entre ambas, en caso de que la hubiera.

3.3. Entrenamiento y evaluación

En esta sección, se aborda en detalle el proceso de entrenamiento seguido. También se describe el proceso de evaluación utilizado para medir el rendimiento del modelo en la tarea de clasificación de relaciones léxico-semánticas. Además, se proporciona una explicación concisa sobre las métricas de evaluación empleadas para analizar los

resultados obtenidos.

3.3.1. Proceso de entrenamiento

En este trabajo, el proceso de entrenamiento se ha llevado a cabo utilizando los modelos de lenguaje RoBERTa *large* y RoBERTa *base*, tanto con *adapters* como sin ellos. Se han realizado 5 iteraciones de entrenamiento y evaluación para cada combinación de modelo, conjunto de datos (EVALution, ROOT09 y CogALex-V) y tipo de entrenamiento. Durante cada iteración, se han registrado el tiempo de entrenamiento, las métricas de evaluación y las relaciones predichas en los conjuntos de datos de test para su análisis.

Antes de pasar los datos al modelo, se ha realizado un procesamiento previo para preparar las parejas de palabras y su relación en un formato adecuado. Cada pareja de palabras y su relación se han verbalizado mediante una plantilla de la forma `‘ <W1> ’ <SEP> ‘ <W2> ’`. En el artículo de Pitarch et al. [24] se demuestra que una plantilla así de sencilla se ha mostrado eficaz para clasificar relaciones léxico-semánticas. Los experimentos realizados en dicho trabajo muestran que estas plantillas mínimas funcionan igual de bien que las más complejas para la tarea de clasificación de relaciones léxico-semánticas, lo que permite reducir el esfuerzo humano y el coste computacional, siguiendo un enfoque neutral al idioma.

Posteriormente, se ha utilizado el tokenizador para la verbalización de las parejas de palabras y sus relaciones en una representación numérica comprensible para el modelo.

Se han empleado 10 épocas en cada entrenamiento, con parámetros específicos establecidos en el Anexo B. Para los conjuntos de datos EVALution y ROOT09, se ha utilizado un conjunto de validación para seleccionar el mejor modelo durante el entrenamiento. El modelo seleccionado se ha utilizado posteriormente para realizar la inferencia en los conjuntos de datos de test.

En el caso del conjunto de datos CogALex-V, que no cuenta con datos de validación, se ha entrenado el modelo utilizando únicamente los datos de entrenamiento. El modelo resultante después de las épocas de entrenamiento se ha utilizado para realizar la inferencia en el conjunto de datos de test.

Se ha calculado el tiempo medio de entrenamiento para cada combinación de modelo, conjunto de datos y tipo de entrenamiento, tomando la media de los tiempos de entrenamiento de las 5 iteraciones. Además, se ha obtenido la media ponderada del F-score como métrica de rendimiento en la clasificación de los datos de test para los conjuntos de datos EVALution y ROOT09. En el caso del conjunto de datos CogALex-V, es importante destacar que la mayoría de sus datos están etiquetados como relaciones aleatorias (aproximadamente tres veces el resto de las relaciones), utilizadas

para intentar despistar al modelo. Los pares de palabras no relacionadas (RANDOM) se consideran como ruido y, por lo tanto, no se toman en cuenta en el cálculo de la media ponderada, siguiendo la recomendación de los autores en [6]. En este sentido, la media ponderada del F-score se ha calculado excluyendo la clase RANDOM.

El entrenamiento y la evaluación han sido realizados en una máquina con un modelo de CPU Intel Xeon de 2.20 GHz y una GPU Tesla T4 de 15 GiB a través de Google Colab, consumiendo más de 20 horas de uso de GPU.

3.3.2. Métricas de evaluación

En la evaluación de modelos de lenguaje, es importante utilizar métricas adecuadas que proporcionen una medida precisa y equilibrada del rendimiento. A continuación se describen dos métricas ampliamente utilizadas: el F-score y la media ponderada.

El F-score es una métrica comúnmente utilizada en tareas de clasificación, que combina las métricas *precision* y *recall*. La *precision* se refiere a la proporción de instancias clasificadas como positivas que son realmente positivas, mientras que el *recall* se refiere a la proporción de instancias positivas que son correctamente identificadas por el modelo. El F-score se calcula utilizando la siguiente fórmula:

$$F = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{\text{precision} + \text{recall}} \tag{3.1}$$

El valor del F-score oscila entre 0 y 1, donde un valor más cercano a 1 indica un mejor rendimiento del modelo en la tarea de clasificación.

Además del F-score, se utiliza la media ponderada para tener en cuenta el desequilibrio de clases en los conjuntos de datos. La media ponderada calcula la media de las métricas de evaluación, asignando mayor peso a las clases con mayor número de instancias. La media ponderada se calcula utilizando la siguiente fórmula:

$$\text{Media ponderada} = \sum_{i=1}^{n} \left(\frac{\text{Instancias de la clase}_i}{\text{Total de instancias}} \right) \cdot \text{Métrica de evaluación}_i \tag{3.2}$$

Para el conjunto de datos CogALex-V, se ha calculado la media ponderada del F-score sin tener en cuenta la clase RANDOM. La fórmula para obtener la media ponderada sin tener en cuenta la clase RANDOM es la siguiente:

$$\text{Media ponderada} = \sum_{i \neq RANDOM}^{n} \left(\frac{\text{Instancias de la clase}_i}{\text{Total de instancias} - \text{Instancias de la clase RANDOM}} \right) \cdot \text{Métrica de evaluación}_i \tag{3.3}$$

Capítulo 4

Resultados

En este capítulo, se presentan y analizan los resultados obtenidos al comparar el uso de *adapters* con el enfoque convencional de ajuste fino en la tarea de clasificación de relaciones léxico-semánticas. Se investiga el rendimiento en términos de F-score y el tiempo de entrenamiento, profundizando en el análisis del impacto que los *adapters* tienen en estos aspectos. Además, se investiga la estrategia de incorporar *adapters* en capas específicas del modelo, proporcionando una visión más completa de su influencia en distintos niveles de representación.

4.1. Desempeño del modelo con y sin *adapters*

A continuación se presentan los resultados promedio obtenidos de los entrenamientos para diferentes modelos (RoBERTa *large* y RoBERTa *base*), conjuntos de datos (EVALution, ROOT09 y CogALexV) y tipos de entrenamiento (con *adapters* y sin *adapters*). Los resultados se basan en la media de los F-scores y tiempos de entrenamiento obtenidos a partir de las 5 iteraciones realizadas, así como su desviación típica (σ). Los resultados se muestran en la Tabla 4.1. Para obtener una visión más detallada de los resultados en términos de *precision* y *recall*, se hace referencia a la tabla ampliada de resultados en el Anexo C (Tabla C.1).

En general, los resultados obtenidos revelan que el empleo de *adapters* puede tener un impacto significativo en el desempeño de los modelos en la clasificación de relaciones léxico-semánticas. Cabe mencionar que que se han obtenido unas desviaciones típicas muy bajas en los resultados, lo que indica una alta consistencia y estabilidad en las mediciones.

Para el modelo RoBERTa *large*, se observa que el uso de *adapters* tiende a mejorar ligeramente el F-score en los conjuntos de datos EVALution y ROOT09, donde los valores son superiores en comparación con el modelo sin *adapters*. En EVALution, el modelo con *adapters* alcanza un F-score promedio de 0.785, mientras que el modelo

Modelo	Dataset	Tipo	F-score		Tiempo	
			Media	σ	Media	σ
RoBERTa *large*	EVALution	Con *adapters*	0.785*	0.003	20min 10s	0.0004
		Sin *adapters*	0.772	0.006	38min 10s	0.001
	ROOT09	Con *adapters*	0.94	0.003	34min 56s	0.0006
		Sin *adapters*	0.935	0.002	58min 54s	0.001
	CogALex-V	Con *adapters*	0.712	0.028	11min 30s	0.0004
		Sin *adapters*	0.741	0.02	22min 52s	0.0002
RoBERTa *base*	EVALution	Con *adapters*	0.721	0.005	6min 12s	0.00007
		Sin *adapters*	0.749*	0.01	10min 47s	0.00007
	ROOT09	Con *adapters*	0.921	0.004	10min 58s	0.0004
		Sin *adapters*	0.929	0.007	17min 49s	0.0001
	CogALex-V	Con *adapters*	0.489	0.029	3min 34s	0.00005
		Sin *adapters*	0.689*	0.021	6min 40s	0.00009

Tabla 4.1: Resultados de los entrenamientos para diferentes modelos, conjuntos de datos y tipos de entrenamiento. Las medias marcadas con * muestran diferencias significativas según el test de Welch entre las medias con y sin *adapters*.

sin *adapters* obtiene un F-score promedio de 0.772. En ROOT09, se obtiene un F-score promedio de 0.94 con *adapters* y 0.935 sin *adapters*. Sin embargo, en el conjunto de datos CogALexV, se observa una ligera disminución en el F-score para el modelo con *adapters* en comparación con el modelo sin *adapters*. El F-score promedio obtenido es de 0.712 con *adapters* y 0.741 sin *adapters*.

Por otro lado, para el modelo RoBERTa *base*, se observa que el desempeño en términos de F-score es ligeramente mejor para el modelo sin *adapters* en los conjuntos de datos EVALution y ROOT09, donde los valores son muy cercanos o iguales entre ambos modelos. Sin embargo, para el conjunto de datos CogALex-V, se aprecia una diferencia de 0.2 en el F-score, donde el modelo sin *adapters* obtiene un resultado mejor.

Es importante destacar que el conjunto de datos CogALex-V presenta menos pares de entrenamiento, y la mayoría de ellos son parejas aleatorias. Esto podría influir en los resultados obtenidos para este conjunto de datos.

Además, para evaluar estadísticamente la significancia de las diferencias observadas en los F-scores entre los modelos con y sin *adapters*, se ha realizado un test de Welch. Este test permite comparar las medias de dos grupos cuando las varianzas pueden ser diferentes. Los resultados de las pruebas reportan valores de p-valor significativamente menores que 0.05 en los casos de EVALution para ambos modelos y CogALex-V para RoBERTa *base*. Estos valores de p-valor indican que las diferencias en los promedios de los F-scores son estadísticamente significativas. Aquellas medias con diferencias significativas se han marcado con un asterisco (*) en la Tabla 4.1. En contraste, no se ha observado evidencia de diferencias significativas en el resto de casos mediante el

test de Welch.

En cuanto al tiempo de entrenamiento, se destaca el hecho de que los modelos con *adapters* logran reducir significativamente el tiempo requerido en comparación con el modelo sin *adapters*. Esta diferencia en el tiempo de entrenamiento se debe a que los *adapters* se añaden a modelos preentrenados, lo cual implica un menor número de parámetros a ajustar durante el proceso de entrenamiento. Al reentrenar únicamente el modelo base, se requiere ajustar todos los parámetros, lo que resulta en un tiempo de entrenamiento mayor.

En resumen, la utilización de *adapters* en la clasificación de relaciones léxico-semánticas se presenta como un enfoque tan competitivo como el proceso de ajuste fino de un modelo, gracias a su desempeño equiparable en términos de resultados. Estos resultados indican que los *adapters* pueden contribuir a mejorar la precisión de la clasificación en casos específicos, mientras que en otros conjuntos de datos se mantiene un nivel comparable de desempeño. Esto resalta la versatilidad y potencial de los *adapters* para adaptarse a diferentes tareas y dominios en el procesamiento del lenguaje natural. Aunque pueda haber una ligera disminución en la precisión en ciertos casos, esta diferencia se ve compensada por el beneficio de un tiempo de entrenamiento más corto y la ventaja adicional de reducir el espacio ocupado por el modelo durante el entrenamiento. Por otro lado, aunque la inclusión de los *adapters* conlleva un aumento en el tiempo de inferencia debido al número adicional de parámetros, este incremento puede considerarse despreciable.

Estas características pueden ser útiles en contextos donde se prioriza la implementación eficiente y ágil de modelos. En algunos casos, se podría considerar aceptable sacrificar una pequeña fracción de la precisión para lograr un ahorro significativo de tiempo durante el entrenamiento y cumplir con las necesidades del sistema o la aplicación. Por ejemplo, en el ámbito de la investigación y la experimentación, se abre la posibilidad de explorar de manera ágil diversas arquitecturas y configuraciones de modelos, lo que agiliza el proceso de encontrar las opciones más adecuadas para tareas específicas.

Además, el uso de *adapters* presenta ventajas adicionales. En primer lugar, no se modifica el modelo original, lo que permite una adaptación rápida a diferentes tareas sin comprometer la estructura base. En segundo lugar, al reducir el número de parámetros entrenables, se facilita la adaptación de modelos más grandes. Por último, en las situaciones en las que el modelo base ya es viable en términos de espacio y almacenamiento, los *adapters* también lo son, dado su impacto mínimo.

4.2. Análisis de los resultados

En esta sección, se ha llevado a cabo un análisis detallado de los resultados obtenidos en la clasificación de relaciones léxico-semánticas utilizando modelos de lenguaje con y sin *adapters*. El objetivo es comprender en profundidad el desempeño y comportamiento de los modelos y evaluar el impacto del uso de *adapters* en la clasificación de relaciones léxico-semánticas. Para ello, se abordan una serie de preguntas y situaciones clave que van a permitir obtener información relevante sobre el desempeño de los modelos. Además, se busca identificar patrones, tendencias y posibles mejoras que puedan guiar futuras investigaciones en el campo del procesamiento del lenguaje natural.

En cada subsección, se presenta y discute una pregunta o situación específica, seguida de los resultados y conclusiones relevantes.

Cabe mencionar que el análisis se ha centrado únicamente en los resultados de RoBERTa *large* debido a su mejor rendimiento en este estudio.

4.2.1. Porcentaje de aciertos promedio por tipo de relación

El objetivo de este análisis es evaluar cuáles son las relaciones léxico-semánticas con los mayores porcentajes de aciertos en promedio. Además, se busca identificar aquellas relaciones que presentan mayores dificultades para ambos modelos, tanto con *adapters* como sin *adapters*, y comparar si coinciden en ambos casos.

Para llevar a cabo este análisis, se ha obtenido para cada tipo de relación el promedio de aciertos en las 5 iteraciones, en el caso de utilizar *adapters* o solo el modelo, y se ha comparado con el número total de instancias de dicha relación.

En la Figura 4.1 se muestran los porcentajes de aciertos de media en las 5 iteraciones de cada relación en el conjunto de datos EVALution, con y sin *adapters*. El orden de las relaciones por mayor porcentaje de aciertos se mantiene consistentemente en ambos casos, tanto en los modelos con *adapters* como en los modelos sin *adapters*. El tipo de relación que más se acierta en media es "HasProperty", con un 89.69 % de aciertos con *adapters* y un 89.94 % solo con el modelo, seguida de las relaciones "Antonym", "HasA", "PartOf", "IsA", "MadeOf" y por último "Synonym", la relación que más dificultades presenta para ambos modelos, con un 65.05 % de aciertos con *adapters* y un 59.86 % sin *adapters*.

Para evaluar si esta baja tasa de aciertos en la clasificación de sinonimia podría estar relacionada con la escasez de datos de entrenamiento, se ha realizado un análisis comparativo con otras relaciones que cuentan con un menor número de ejemplos disponibles. Se ha observado que tres relaciones, concretamente "MadeOf", "HasA" y "PartOf", tienen un menor número de datos de entrenamiento en comparación, y

aún así, logran un porcentaje de aciertos mayor. Esto resalta la complejidad en la clasificación de la relación de sinonimia.

Se ha aplicado un test de Welch para determinar si hay diferencias estadísticamente significativas entre las medias de aciertos para los distintos tipos de relaciones en los modelos con y sin *adapters*. Los resultados indican p-valores de 0.0591, 0.06862 y 0.04997 para las relaciones "Synonym", "HasA" y "Antonym", respectivamente. Aunque los p-valores están cerca del umbral de significación (0.05), no se pueden establecer diferencias significativas en los aciertos promedio para estas relaciones específicas. Para el resto de relaciones, se han obtenidos unos p-valores elevados (superiores a 0.7).

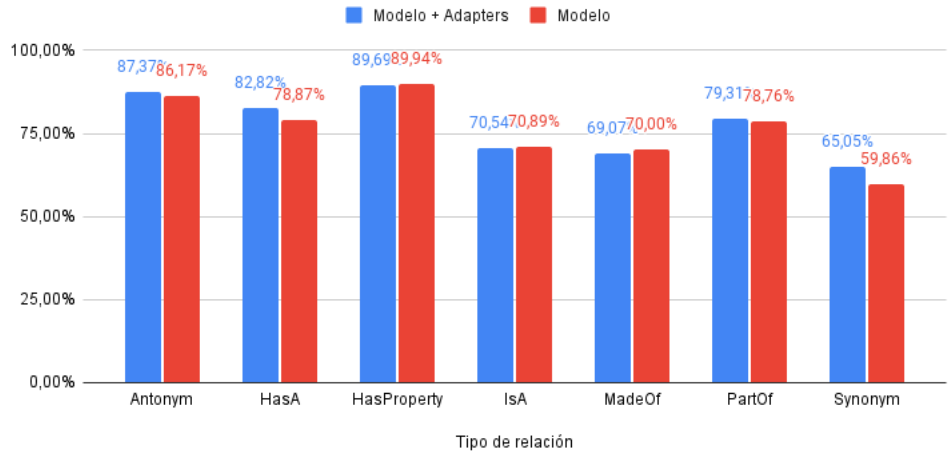


Figura 4.1: Porcentaje promedio de aciertos en EVALution.

En la Figura 4.2 se muestran los porcentajes de aciertos de media en las 5 iteraciones de cada relación en el conjunto de datos ROOT09, con y sin *adapters*. El orden de las relaciones por mayor porcentaje de aciertos se mantiene consistentemente en ambos casos, tanto en los modelos con *adapters* como en los modelos sin *adapters*. La relación con mayor porcentaje de aciertos es la de cohipónimos ("CORD") con un 99.51 % con *adapters* y un 99.26 % solo con el modelo. Le siguen las parejas son ningún tipo de relación ("RANDOM") y la relación "HYPER", ambas por encima del 90 % en cada tipo de entrenamiento.

En la Figura 4.3 se muestran los porcentajes de aciertos de media en las 5 iteraciones de cada relación en el conjunto de datos CogALex-V, con y sin *adapters*. El orden de las relaciones por mayor porcentaje de aciertos se mantiene consistentemente en ambos casos, tanto en los modelos con *adapters* como en los modelos sin *adapters*. La relación con mayor porcentaje de aciertos es la de antónimos ("ANT") con un 86.89 % con *adapters* y un 86.28 % solo con el modelo. Le siguen las relaciones "PART_OF", "HYPER" y por últimos los sinónimos ("SYN").

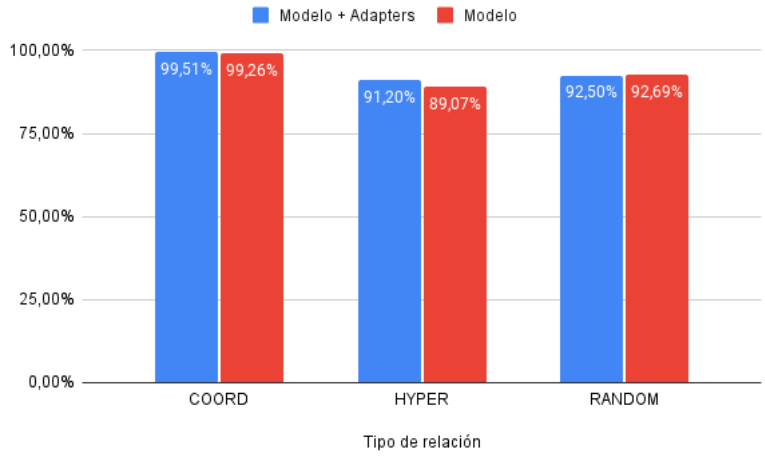


Figura 4.2: Porcentaje promedio de aciertos en ROOT09.

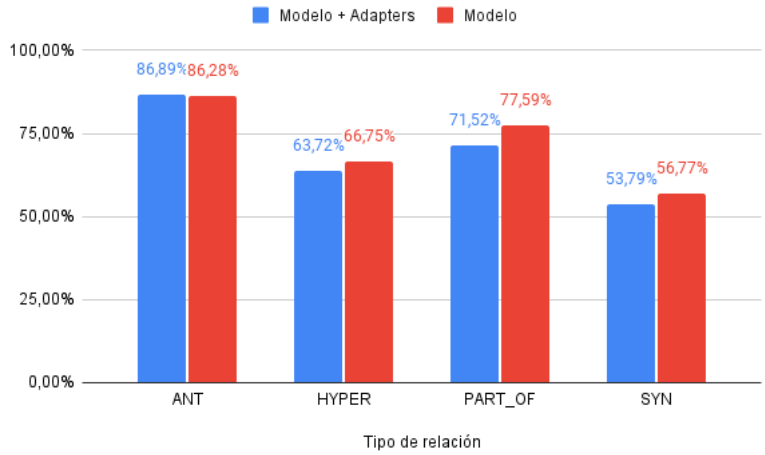


Figura 4.3: Porcentaje promedio de aciertos en CogALex-V.

Siguiendo el mismo enfoque que para las tasas de acierto en EVALution, se ha llevado a cabo un análisis similar para las tasas de acierto en CogALex-V. Al igual que en el caso anterior, se han comparado las relaciones con un menor número de datos de entrenamiento. Se ha observado que la relación "PART_OF", a pesar de tener menos ejemplos disponibles que la sinonimia, logra alcanzar un porcentaje de aciertos superior.

Se puede concluir que el tipo de relación léxico-semántica que más dificultad presenta son los sinónimos, lo que podría deberse a la sutil diferencia de significado entre estas palabras. Por otro lado, los antónimos, al presentar significados opuestos, pueden ser más fácilmente distinguidos por los modelos, lo que se refleja en sus altos porcentajes de acierto.

4.2.2. Porcentaje de aciertos por tipo de relación en todas las iteraciones

El objetivo de este análisis consiste en analizar las parejas de datos cuya relación léxico-semántica ha sido acertada en todas las iteraciones. El objetivo es determinar cuáles relaciones presentan una alta frecuencia de aciertos y cuáles tienen un rendimiento más bajo en ambos modelos. Además, se busca evaluar si la inclusión de *adapters* influye en los resultados, generando errores de naturaleza distinta o si, por el contrario, los errores son consistentes entre ambos enfoques.

Para llevar a cabo este análisis, se han obtenido para cada tipo de relación aquellas parejas de datos cuya relación ha sido clasificada correctamente en las 5 iteraciones, en el caso de utilizar *adapters* o solo el modelo, y se ha comparado con el número total de instancias de dicha relación.

En la Figura 4.4 se muestran los porcentajes de aciertos de cada tipo de relación, teniendo en cuenta aquellas parejas de palabras cuya relación ha sido clasificada correctamente en todas las iteraciones, en el conjunto de datos EVALution, con y sin *adapters*. Al igual que en el análisis anterior, la relación con mayor porcentaje de aciertos es "HasProperty", con un 86.02 % con *adapters* y un 85.09 % sin *adapters*. También coincide la relación con peores resultados, "Synonym", con porcentajes inferiores al 50 % para cada tipo de entrenamiento. Esto muestra que menos del 50 % de las parejas de datos con relación de sinonimia han sido clasificadas correctamente en todas las iteraciones para cada tipo de entrenamiento.

Se observa un mejor rendimiento en todas las relaciones al emplear *adapters*.

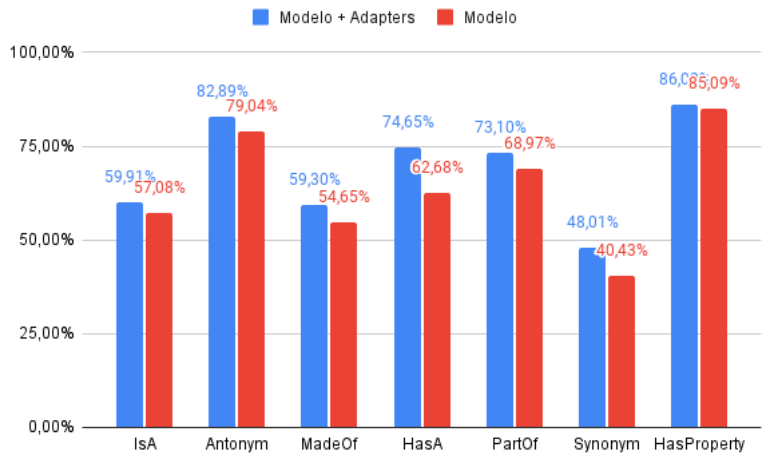


Figura 4.4: Porcentaje de aciertos por relación en EVALution.

En la Figura 4.5 se muestran los porcentajes de aciertos de cada tipo de relación, teniendo en cuenta aquellas parejas de palabras cuya relación ha sido clasificada

correctamente en todas las iteraciones, en el conjunto de datos ROOT09, con y sin *adapters*. El orden de aciertos de cada tipo de relación coincide con el del anterior análisis.

Se observa un mejor rendimiento en todas las relaciones al emplear *adapters*.

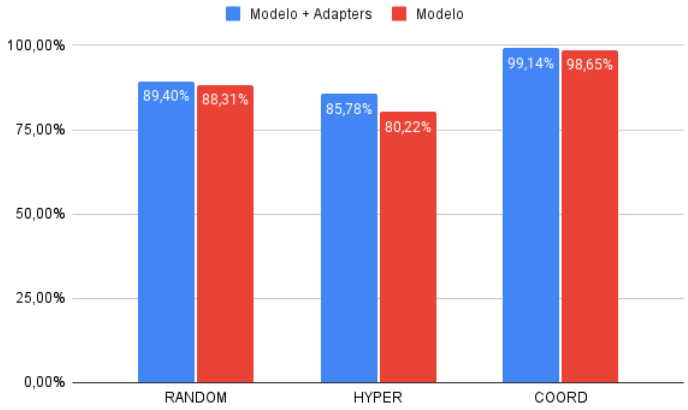


Figura 4.5: Porcentaje de aciertos por relación en ROOT09.

En la Figura 4.6 se muestran los porcentajes de aciertos de cada tipo de relación, teniendo en cuenta aquellas parejas de palabras cuya relación ha sido clasificada correctamente en todas las iteraciones, en el conjunto de datos CogALex-V, con y sin *adapters*. Como en el anterior análisis, la relación con mayor porcentaje de aciertos son los antónimos ("ANT"). El porcentaje de aciertos con *adapters* es un 80 % y sin *adapters* un 73.06 %. La relación que presenta mayor dificultad es la sinonimia ("SYN"), con un porcentaje de aciertos muy bajo.

Se puede observar que los *adapters* empeoran el rendimiento de clasificación de esta relación léxico-semántica, así como la relación "PART_OF".

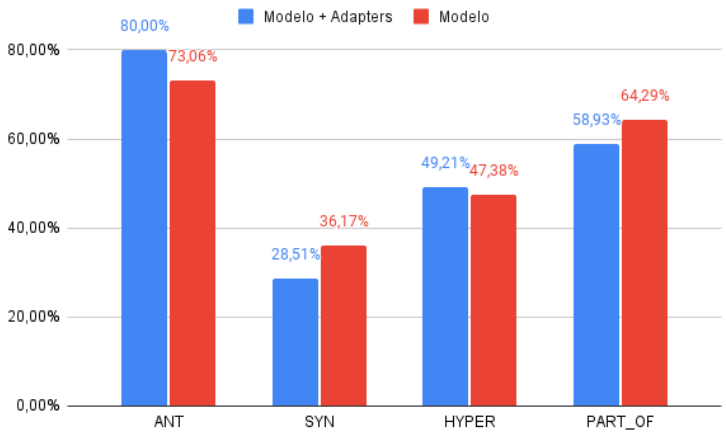


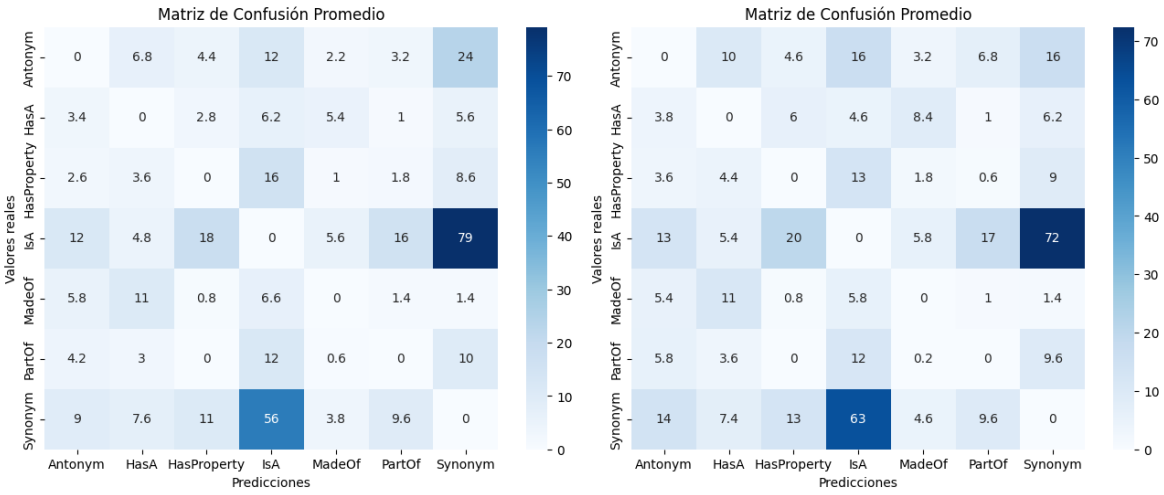
Figura 4.6: Porcentaje de aciertos por relación en CogALex-V.

4.2.3. Patrones de fallos promedio

El objetivo de este análisis es identificar patrones de fallos que se presentan en promedio en las cinco iteraciones realizadas, con el fin de identificar los errores más comunes cometidos por los modelos y evaluar si estos errores de confusión son específicos de un modelo en particular (con o sin *adapters*) o si se presentan en ambos casos.

Para llevar a cabo este análisis, se han calculado las matrices promedio de confusión de las 5 iteraciones que contienen los errores de clasificación para cada clase en cada conjunto de datos, tanto utilizando *adapters* como únicamente el modelo. En estas matrices, las diagonales presentan un valor de cero debido a que solo se están considerando los fallos de clasificación, excluyendo los aciertos.

En la Figura 4.7 se muestran las matrices de confusión promedio para el conjunto de datos EVALution, tanto con como sin *adapters*. Se destaca que las relaciones que experimentan la mayor confusión son "IsA" y "Synonym", en ambos casos. Con *adapters*, "IsA" se confunde con "Synonym" en un 58.58 % de las instancias, mientras que al revés, "Synonym" se confunde con "IsA" en un 58.06 % de los casos. Sin la utilización de *adapters*, los porcentajes de confusión son del 54.19 % para "IsA" confundido con "Synonym" y del 56.83 % para "Synonym" confundido con "IsA".



(a) Con *adapters*.

(b) Sin *adapters*.

Figura 4.7: Matrices de confusión promedio de fallos con EVALution.

En la Figura 4.8 se muestran las matrices de confusión promedio para el conjunto de datos ROOT09, tanto con como sin *adapters*. No se ha identificado ningún patrón relevante.

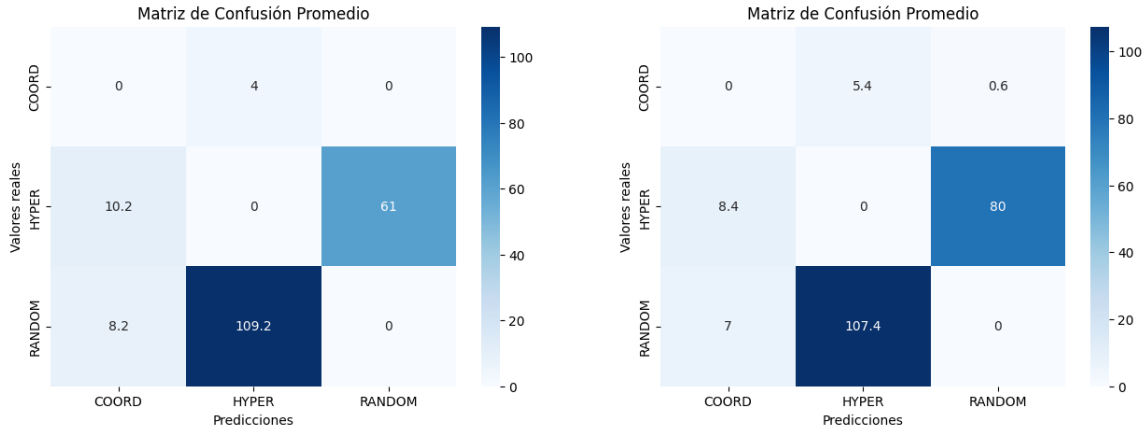
En la Figura 4.9 se muestran las matrices de confusión promedio para el conjunto de datos CogAlex-V, tanto con como sin *adapters*. En este caso, se puede observar que las

relaciones que más se confunden entre sí son hiperónimos y sinónimos, en ambos casos. Con *adapters*, "HYPER"se confunde con "SYN" en un 56.27 % de los casos, y viceversa, "SYN" se confunde con "HYPER" en un 55.06 % de las instancias. En contraste, sin la utilización de *adapters*, los porcentajes de confusión son del 48.03 % para "HYPER" confundido con "SYN" y del 55.11 % para "SYN" confundido con "HYPER".

En los tres conjuntos de datos, las matrices de confusión promedio son parecidas en ambos enfoques, lo que sugiere que los errores son similares entre ellos.

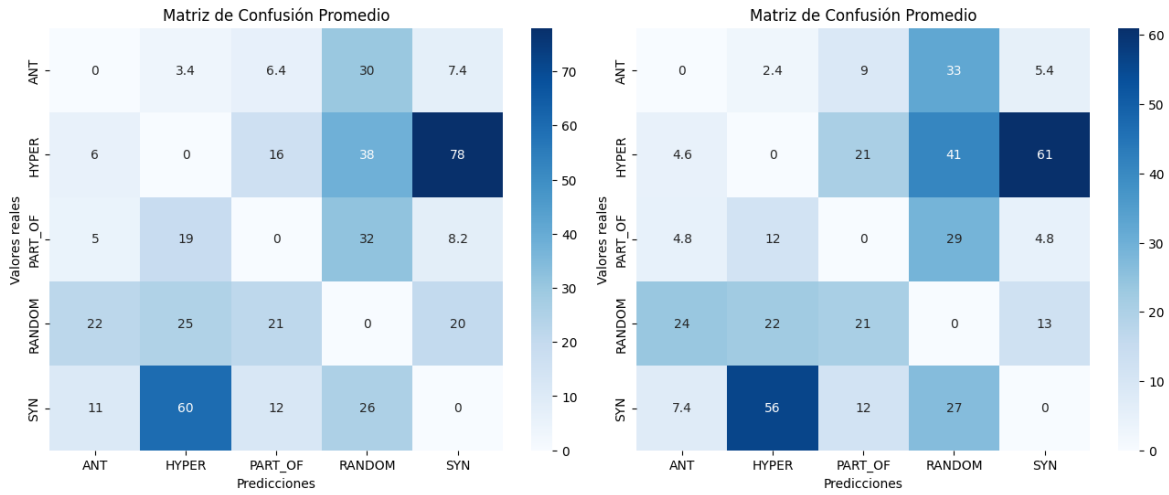
4.2.4. Patrones de fallos recurrentes en todas las iteraciones

El objetivo de este análisis es examinar si existen patrones consistentes en la clasificación errónea de relaciones, tanto con *adapters* como sin ellos, en todas las ejecuciones realizadas.



(a) Con *adapters*. (b) Sin *adapters*.

Figura 4.8: Matrices de confusión promedio de fallos con ROOT09.



(a) Con *adapters*. (b) Sin *adapters*.

Figura 4.9: Matrices de confusión promedio de fallos con CogALex-V.

Para llevar a cabo este análisis, se han calculado las matrices de confusión respecto a aquellas parejas de datos cuya relación ha sido clasificada erróneamente en las 5 iteraciones y coinciden en ambos tipos de entrenamiento, es decir, han sido erróneamente clasificadas en las 5 iteraciones con *adapters* y han sido erróneamente clasificadas en las 5 iteraciones solo con el modelo.

En las tres figuras siguientes, se presenta la matriz de confusión, junto con dos tablas complementarias. La primera tabla muestra la distribución de los errores de clasificación de cada clase, es decir, el número de parejas mal clasificadas en las 5 iteraciones, tanto con *adapters* como sin *adapters*, y que coinciden en ambos casos. Esta distribución se relaciona con el número total de parejas con dicha relación en el conjunto de datos. La segunda tabla muestra los porcentajes de confusión de clasificación de cada clase en comparación con el resto, donde las filas representan la etiqueta real y las columnas la etiqueta predicha.

Para el conjunto de datos EVALution (ver Figura 4.10, Tabla 4.2), se obtiene que "IsA" es la relación que más cuesta de clasificar, con un 11.55 % de parejas mal clasificadas del total de parejas con dicha relación. Además, en la Tabla 4.3 se puede observar como las relaciones que más se confunden son "IsA" con "Synonym" en un 62.26 % y al revés en un 57.69 %. Destacan otros patrones como que la relación "PartOf" se clasifica como "Synonym" en un 46.15 % y la relación "HasProp" con "IsA" en un 46.15 %.

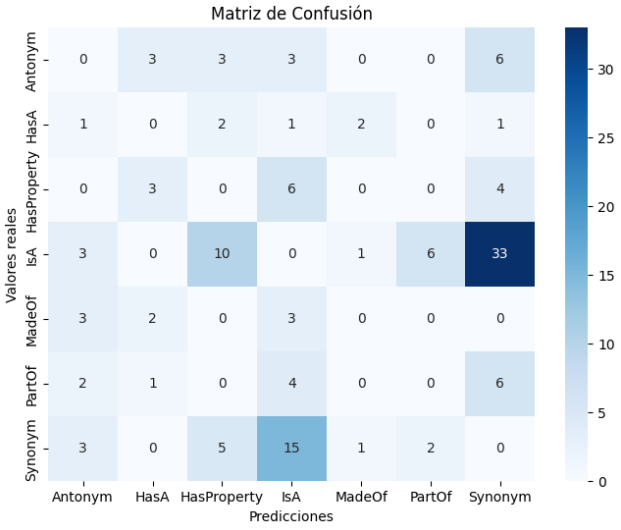


Figura 4.10: Patrónes de fallos en EVALution.

Para el conjunto de datos ROOT09 (ver Figura 4.11, Tabla 4.4, Tabla 4.5), no se ha identificado ningún patrón significativo en los errores de clasificación, ya que este conjunto muestra un buen rendimiento general en la clasificación. Es importante destacar que el 0 % de errores relativos a la clase "COORD" indica que todas las parejas

Relación	Fallos	Porcentaje
IsA	53	11.55 %
Antonym	15	3.61 %
MadeOf	8	9.30 %
HasA	7	4.93 %
PartOf	13	8.97 %
Synonym	26	9.39 %
HasProperty	13	4.04 %

Tabla 4.2: Distribución de los errores de cada clase en EVALution.

	IsA	Antonym	MadeOf	HasA	PartOf	Synonym	HasProp
IsA	0.00 %	5.66 %	1.89 %	0.00 %	11.32 %	62.26 %	18.87 %
Antonym	20.00 %	0.00 %	0.00 %	20.00 %	0.00 %	40.00 %	20.00 %
MadeOf	37.50 %	37.50 %	0.00 %	25.00 %	0.00 %	0.00 %	0.00 %
HasA	14.29 %	14.29 %	28.57 %	0.00 %	0.00 %	14.29 %	28.57 %
PartOf	30.77 %	15.38 %	0.00 %	7.69 %	0.00 %	46.15 %	0.00 %
Synonym	57.69 %	11.54 %	3.85 %	0.00 %	7.69 %	0.00 %	19.23 %
HasProp	46.15 %	0.00 %	0.00 %	23.08 %	0.00 %	30.77 %	0.00 %

Tabla 4.3: Porcentajes de confusión de cada clase en EVALution.

de datos en el conjunto de pruebas se clasifican correctamente en al menos una de las cinco iteraciones realizadas.

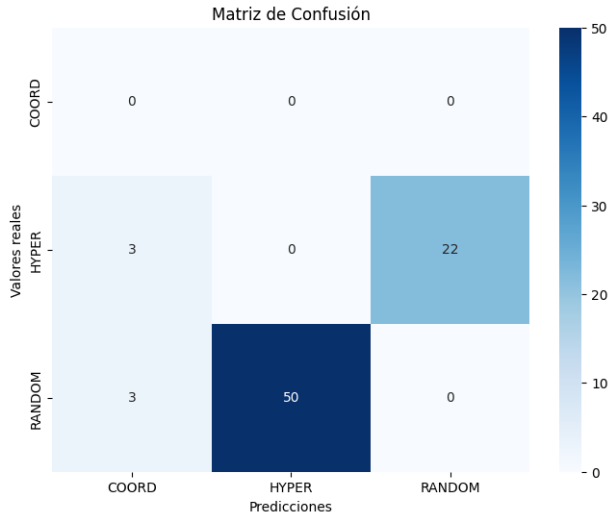


Figura 4.11: Patrónes de fallos en ROOT09.

Relación	Fallos	Porcentaje
RANDOM	53	3.38 %
HYPER	25	3.09 %
COORD	0	0.00 %

Tabla 4.4: Distribución de los errores de cada clase en ROOT09.

	RANDOM	HYPER	COORD
RANDOM	0.00 %	94.34 %	5,66 %
HYPER	88.00 %	0.00 %	12,00 %
COORD	0.00 %	0.00 %	0.00 %

Tabla 4.5: Porcentajes de confusión de cada clase en ROOT09.

Para el conjunto de datos CogALex-V (ver Figura 4.12, Tabla 4.6, Tabla 4.7), se observa que la relación de sinonimia presenta la tasa más alta de errores relativos, alcanzando un 8.94 %. Además, esta relación tiende a confundirse con la hiperonimia en un 38.10 % de los casos, y la hiperonimia, a su vez, se confunde con la sinonimia en un 54.55 % de las ocasiones. Es importante mencionar que todas las relaciones muestran un porcentaje considerable de errores clasificados como "RANDOM", lo cual es comprensible debido a la naturaleza del conjunto de datos, donde es más propenso a cometer errores en este tipo de relaciones.

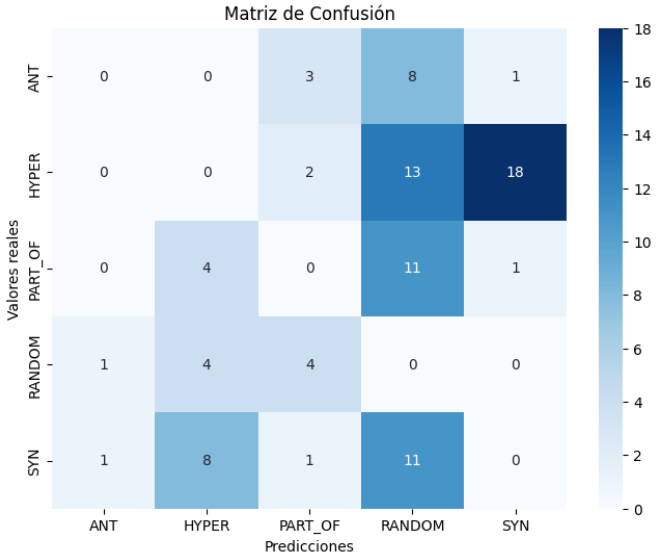


Figura 4.12: Patrónes de fallos en CogALex-V.

Relación	Fallos	Porcentaje
ANT	12	3.33 %
SYN	21	8.94 %
HYPER	33	8.64 %
PART_OF	16	7.14 %
RANDOM	9	0.29 %

Tabla 4.6: Distribución de los errores de cada clase en CogALex-V.

	ANT	SYN	HYPER	PART_OF	RANDOM
ANT	0.00 %	8.33 %	0.00 %	25.00 %	66.67 %
SYN	4.76 %	0.00 %	38.10 %	4.76 %	52.38 %
HYPER	0.00 %	54.55 %	0.00 %	6.06 %	39.39 %
PART_OF	0.00 %	6.25 %	25.00 %	0.00 %	68.75 %
RANDOM	11.11 %	0.00 %	44.44 %	44.44 %	0.00 %

Tabla 4.7: Porcentajes de confusión de cada clase en CogALex-V.

4.2.5. Grado de solape entre relaciones mal clasificadas

El objetivo de este análisis es medir el grado de solape entre las relaciones que han sido mal clasificadas tanto con *adapters* como sin *adapters* entre sí. Esto permitirá comprender si hay similitudes o diferencias en las relaciones que son incorrectamente clasificadas en ambos enfoques.

El grado de solape entre las relaciones mal clasificadas con y sin *adapters* se ha calculado identificando aquellas relaciones que han sido mal clasificadas en todas las iteraciones para ambos casos. Se ha obtenido la intersección de estas relaciones falladas y se ha divido entre la unión de todas las relaciones falladas, lo que proporciona una medida de similitud entre las clasificaciones erróneas.

En el dataset EVALution, el 51,53 % de las relaciones mal clasificadas se superponen entre ambos modelos, mientras que en el dataset ROOT09, este solape es del 56,52 %. En el caso del dataset CogALex-V, el solape es menor, con un 40,43 %. Estos resultados sugieren que los modelos con y sin *adapters* pueden compartir características similares en la clasificación errónea de relaciones léxico-semánticas en cierto conjunto de relaciones, mientras que otras relaciones son más sensibles a las particularidades de cada enfoque.

En la Tabla 4.8 se presentan cinco ejemplos de relaciones del conjunto de datos EVALution que han sido incorrectamente clasificadas tanto por los modelos con *adapters* como sin ellos. Estos errores se han destacado para enfatizar que incluso un humano podría caer en la misma equivocación de clasificación, lo que sugiere que los resultados obtenidos representan un umbral inferior y que posiblemente los resultados reales sean mejores de lo que indican. Estos ejemplos exponen que algunas de las anotaciones originales resultan engañosas y que el modelo predice relaciones más coherentes.

Estos ejemplos ilustran cómo las múltiples acepciones y matices semánticos de una palabra pueden dificultar la clasificación precisa de relaciones léxico-semánticas, tanto para los modelos como para los humanos.

Pareja	Anotación	Predicción
(cloth, cotton)	HasA	MadeOf
(england, great britain)	IsA	PartOf
(fish, animal)	Antonym	IsA
(orange, fruit)	PartOf	IsA
(rock, hard)	IsA	HasProperty

Tabla 4.8: Ejemplos de parejas falladas por ambos modelos.

4.3. Rendimiento por capas con *adapters*

Si bien el entrenamiento completo con *adapters* en todas las capas podría apuntar a un rendimiento óptimo, esta estrategia también conlleva un aumento en la cantidad de parámetros entrenados, lo que podría extender los tiempos de ajuste fino del modelo.

Una alternativa de interés es dirigir el entrenamiento hacia un subconjunto específico de capas. Sin embargo, es importante señalar que esta estrategia también podría implicar un ligero compromiso en el rendimiento en comparación con el entrenamiento completo. Al limitar el ajuste a un subconjunto, se reduce la cantidad de parámetros involucrados y, por tanto, los tiempos de entrenamiento.

En esta sección, se analiza el rendimiento de los modelos con respecto a la adición de *adapters* en capas específicas del modelo. Para llevar a cabo este análisis, se ha realizado un total de 5 iteraciones de entrenamiento y evaluación para cada conjunto de datos, modelo y conjunto de capas. Se distinguen tres conjuntos de capas para cada modelo: primeras capas, capas intermedias y últimas capas. Cada uno de estos conjuntos abarca un tercio del total de capas del respectivo modelo. Cabe destacar que en el caso de RoBERTa *base*, se consideran las capas 0 a 3 como primeras capas, las capas 4 a 7 como capas intermedias y las capas 8 a 11 como últimas capas. En el caso de RoBERTa *large*, las capas 0 a 7 corresponden a primeras capas, las capas 8 a 15 a capas intermedias y las capas 16 a 23 a últimas capas, dado que este modelo cuenta con 24 capas en total.

Durante cada una de las iteraciones, se han registrado las métricas de evaluación y las relaciones predichas en los conjuntos de datos de prueba.

A continuación, se presentan los resultados de los entrenamientos con la incorporación de *adapters* en diferentes conjuntos de capas de los modelos RoBERTa *large* y RoBERTa *base*. Además, se han añadido los resultados de incorporar los *adapters* en todas las capas de los modelos, los cuales se han presentado anteriormente en la Tabla 4.1. La Tabla 4.9 muestra el rendimiento de los modelos en términos de F-score, junto con la desviación estándar (σ) correspondiente para cada conjunto de capas y conjunto de datos evaluados.

Dataset	Capas	RoBERTa *large*		RoBERTa *base*	
		F-score	σ	F-score	σ
EVALution	Prim.	0.63	0.033	0.605	0.015
	Inter.	0.751	0.004	0.668	0.01
	Ult.	0.703	0.012	0.566	0.004
	Todas	0.785	0.003	0.721	0.005
ROOT09	Prim.	0.935	0.004	0.915	0.004
	Inter.	0.922	0.003	0.887	0.002
	Ult.	0.91	0.003	0.846	0.003
	Todas	0.94	0.003	0.921	0.004
CogALex-V	Prim.	0.303	0.02	0.311	0.006
	Inter.	0.579	0.031	0.387	0.017
	Ult.	0.492	0.011	0.278	0.024
	Todas	0.712	0.028	0.489	0.029

Tabla 4.9: Resultados de los entrenamientos con *adapters* en diferentes capas.

En la tabla presentada, se destacan ciertas tendencias en el rendimiento de los modelos al entrenarlos con *adapters* en diferentes conjuntos de capas. En particular, se observa que los resultados varían dependiendo del conjunto de datos y el modelo evaluado.

Para los conjuntos de datos EVALution y CogALex-V, los resultados más sobresalientes se obtienen al entrenar los modelos con *adapters* en las capas intermedias.

En el caso de ROOT09, se observa un patrón diferente. Aquí, las primeras capas muestran el mejor rendimiento al emplear *adapters*. Es importante destacar que en este caso, el rendimiento de las primeras capas no es significativamente inferior al entrenamiento con *adapters* en todas las capas.

Además de las tendencias observadas en este estudio, sería interesante llevar a cabo un análisis más detallado en futuras investigaciones. Se podrían explorar diversas estrategias de combinación de capas, como entrenar subconjuntos de capas en combinación, para determinar si ciertas configuraciones de capas pueden interactuar de manera complementaria y así mejorar aún más el rendimiento de los modelos con *adapters*.

Capítulo 5

Conclusiones y trabajo futuro

Los resultados obtenidos en este estudio sugieren que los *adapters* son un método tan competitivo como el ajuste fino convencional de un modelo, ya que ambas aproximaciones ofrecen un desempeño similar. La ventaja clave de los *adapters* radica en la eficiencia, ya que solo requieren entrenar los parámetros adicionales, lo que resulta en un tiempo de entrenamiento menor en comparación con el ajuste fino completo del modelo. Además, este enfoque contribuye a reducir el espacio ocupado por el modelo durante el entrenamiento.

En este sentido, se ha encontrado un balance interesante entre el rendimiento y el tiempo de entrenamiento al usar *adapters*. Los *adapters* ofrecen una ventaja significativa al acelerar la adaptación del modelo a tareas específicas, lo cual puede ser útil en situaciones donde la implementación eficiente y ágil de modelos es clave. Por ejemplo, pueden facilitar la exploración ágil de arquitecturas y configuraciones en en el ámbito de la investigación y la experimentación. Estos permiten una adaptación rápida a diferentes tareas sin alterar la estructura base del modelo. Además, reducen el número de parámetros entrenables, facilitando la adaptación de modelos más grandes. En situaciones donde el modelo base es viable en términos de espacio, los *adapters* también lo son, dado su impacto mínimo.

La sinonimia se ha identificado como la relación léxico-semántica con mayor dificultad para clasificarse correctamente. Además, esta relación tiende a confundirse con la hiperonimia.

Para futuras investigaciones, se plantea explorar la optimización de los hiperparámetros y la evaluación de *adapters* en otros conjuntos de datos de clasificación de relaciones léxico-semánticas. También se sugiere realizar análisis más detallados de los errores cometidos por los modelos para identificar posibles mejoras y patrones de dificultad en la clasificación. Además, sería beneficioso investigar en qué capas del modelo se encuentran las características más relevantes para la tarea de clasificación y probar diferentes plantillas de verbalización.

Capítulo 6

Bibliografía

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[4] Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, pages 64–69, Beijing, China, July 2015. Association for Computational Linguistics.

[5] Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4557–4564, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[6] Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. The CogALex-V shared task on the corpus-based identification of semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 69–79, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[7] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020.

[8] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018.

[11] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[12] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*, 2020.

[13] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[14] Himashi Rathnayake, Janani Sumanapala, Raveesha Rukshani, and Surangika Ranathunga. Adapter-based fine-tuning of pre-trained multilingual language models for code-mixed and code-switched text classification. *Knowledge and Information Systems*, 64, 07 2022.

[15] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992.

[16] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[19] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[20] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

[21] Hugo Liu and Push Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22, 06 2004.

[22] Giulia Benotto. *Distributional models for semantic relations: A sudy on hyponymy and antonymy*. PhD thesis, University of Pisa, 2015.

[23] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, 2011.

[24] Lucia Pitarch, Jordi Bernad, Lacramioara Dranca, Carlos Bobed Lisbona, and Jorge Gracia. No clues good clues: out of context lexical relation classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5607–5625, Toronto, Canada, July 2023. Association for Computational Linguistics.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Repositorio de código

URI: https://github.com/jbuil/LexSem-Adapters
Licencia: MIT
Estructura del repositorio:

```
LexSem-Adapters/
|-- notebooks/
|   |-- LexSem-Adapters.ipynb
|-- results/
|   |-- results.xlsx
|-- datasets
|   |-- lexical_datasets.zip
|-- scripts
|   |-- lrc_train_evaluate_adapters.py
|   |-- welch.R
|-- experiments/
|   |-- main/
|   |   |-- EVALution/
|   |   |   |-- roberta-large/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |   |   |-- roberta-base/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
```

```
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |   |-- ROOT09/
|   |   |   |-- roberta-large/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |   |   |-- roberta-base/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |   |-- CogALex-V/
|   |   |   |-- roberta-large/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |   |   |-- roberta-base/
|   |   |   |   |-- adapters
|   |   |   |   |   |-- ...
|   |   |   |   |-- model
|   |   |   |   |   |-- ...
|   |   |   |   |-- analysis
|   |   |   |   |   |-- ...
|   |-- layers/
|   |   |-- EVALution/
|   |   |   |-- roberta-large/
|   |   |   |   |-- ...
```

```
|   |   |   |-- roberta-base/
|   |   |   |   |-- ...
|   |   |-- ROOT09/
|   |   |   |-- roberta-large/
|   |   |   |   |-- ...
|   |   |   |-- roberta-base/
|   |   |   |   |-- ...
|   |   |-- CogALex-V/
|   |   |   |-- roberta-large/
|   |   |   |   |-- ...
|   |   |   |-- roberta-base/
|   |   |   |   |-- ...
|-- README.md
|-- LICENSE
```

Anexos B

Parámetros de entrenamiento

Tipo	Dataset	Épocas	Tamaño lote	LR	Optimizador
Adapters	EVAL, R09	10	32/64	1e-4	AdamW ($\beta_1 = 0{,}9, \beta_2 = 0{,}999$)
Modelo	EVAL, R09	10	32/64	2e-5	AdamW ($\beta_1 = 0{,}9, \beta_2 = 0{,}999$)
Adapters	CogALex-V	10	32/-	1e-4	AdamW ($\beta_1 = 0{,}9, \beta_2 = 0{,}999$)
Modelo	CogALex-V	10	32/-	2e-5	AdamW ($\beta_1 = 0{,}9, \beta_2 = 0{,}999$)

Tabla B.1: Parámetros empleados en el entrenamiento con diferentes conjuntos de datos.

Para EVALution y ROOT09, que ambos disponen de un conjunto de validación, se ha utilizado el F-score como métrica de validación. Además, se ha empleado el mecanismo para guardar el mejor modelo al final del entrenamiento, es decir, aquel modelo que obtenga la mejor métrica en el conjunto de validación en cada época.

Anexos C

Tabla ampliada de resultados

Modelo	Dataset	Tipo	Precision	Recall	F-score		Tiempo	
					Media	σ	Media	σ
RoBERTa *large*	EVALution	Con *adapters*	0.788	0.784	0.785*	0.003	20min 10s	0.0004
		Sin *adapters*	0.773	0.772	0.772	0.006	38min 10s	0.001
	ROOT09	Con *adapters*	0.941	0.94	0.94	0.003	34min 56s	0.0006
		Sin *adapters*	0.935	0.935	0.935	0.002	58min 54s	0.001
	CogALex-V	Con *adapters*	0.725	0.702	0.712	0.028	11min 30s	0.0004
		Sin *adapters*	0.757	0.727	0.741	0.02	22min 52s	0.0002
RoBERTa *base*	EVALution	Con *adapters*	0.722	0.722	0.721	0.005	6min 12s	0.00007
		Sin *adapters*	0.751	0.75	0.749*	0.01	10min 47s	0.00007
	ROOT09	Con *adapters*	0.923	0.92	0.921	0.004	10min 58s	0.0004
		Sin *adapters*	0.929	0.929	0.929	0.007	17min 49s	0.0001
	CogALex-V	Con *adapters*	0.52	0.489	0.489	0.029	3min 34s	0.00005
		Sin *adapters*	0.704	0.677	0.689*	0.021	6min 40s	0.00009

Tabla C.1: Resultados ampliados de la Tabla 4.1.