CIATEQ, A. C. Centro de Tecnología Avanzada
Gerencia de Posgrado


# *Automatización de rastreabilidad de requerimientos de software por medio de procesamiento de lenguaje natural*

TESIS QUE PRESENTA

**Mtro. Carlos Octavio Mendívil Vázquez**
**Asesor: Dr. Luis Ricardo Corral Velázquez**

Para obtener el grado de

**Doctor en**
**Manufactura Avanzada**

Querétaro, Querétaro
enero, 2024

GE Aerospace

Campo Real 1692
Residencial El Refugio
Querétaro, Qro., 7616.

Querétaro, Qro. 6 de febrero de 2024

**CIATEQ, A.C., CENTRO DE TECNOLOGÍA AVANZADA**
Avenida del Retablo, número 150,
Col. FOVISSSTE, C.P. 76150,
Santiago de Querétaro, Qro.,
**Asunto:** Manejo de información confidencial

A quien corresponda,

Por medio de la presente carta, se hace constar que el trabajo de investigación, identificado bajo el título de tesis *"Automatización de rastreabilidad de requerimientos de software por medio de procesamiento de lenguaje natural"*, elaborado por el empleado **Carlos Octavio Mendivil Vazquez**, para la obtención del grado **Doctorado en Manufactura Avanzada** con la Institución **CIATEQ, A.C.,** fue realizado bajo el marco del Convenio General de Colaboración Científica y Tecnológica celebrado entre CENTRO DE INGENIERÍA AVANZADA EN TURBOMAQUINAS, S. DE R.L. DE C.V. ("GEIQ"), y CIATEQ, A.C. ("CIATEQ"), con fecha 10 de mayo de 2018.

Se expide la presente carta bajo solicitud de CIATEQ para los fines que esta determine.

Atentamente,

CENTRO DE INGENIERÍA AVANZADA
EN TURBOMAQUINAS, S. DE R. L. DE C. V.
RFC. CIA-990714-TF6
Campo Real 1692, Residencial el Refugio
Querétaro, Qro. C. P. 76146
Tel. ( 442 ) 296 23 00, Fax. ( 442 ) 296 23 10

Líder de Educación GEIQ

# CARTA DE LIBERACIÓN DEL ASESOR

GE
Infrastructure Querétaro

Campo Real 1692
Ampliacion El Refugio C.P. 76146
Santiago de Querétaro, Qro. México

T +52 (442) 296 2300
F +52 (442) 296 2310

**Fecha:** 1 de Septiembre de 2023

Mtro. Geovany González Carlos
Gerencia de Posgrado
CIATEQ, A.C.

El abajo firmantes, miembro del Comité Tutorial del grado y nombre completo del alumno **Carlos Octavio Mendivil Vázquez**, una vez revisado su Proyecto Terminal de Tesis Doctoral, titulado "**Automatización de rastreabilidad de requerimientos de software por medio de NLP**" autorizo que el citado trabajo sea presentado por el alumno para su revisión, con el fin de alcanzar el grado de **Doctorado en Manufactura Avanzada**.

Sin otro particular por el momento, agradezco la atención prestada.

**Dr. Luis Ricardo Corral Velazquez**
Asesor
Edison and Learning Leader
Luis.Corral@ge.com

CENTRO DE INGENIERÍA AVANZADA EN TURBOMAQUINAS, S. DE R. L. DE C. V.
RFC. CIA-990714-TF6
Campo Real 1692, Residencial el Refugio
Querétaro, Qro. C. P. 76146
Tel. ( 442 ) 296 23 00, Fax. ( 442 ) 296 23 10

# CARTA DE LIBERACIÓN DEL REVISOR

Pachuca de Soto, Hidalgo, 26 de enero del 2024.

Mtro. Geovany González Carlos
Gerencia de Posgrado
CIATEQ, A.C.

Por medio de la presente me dirijo a usted en calidad de Revisora del proyecto terminal del (la) alumno (a) **Mtro. Carlos Octavio Mendívil Vázquez**, cuyo título es:

**"Automatización de rastreabilidad de requerimientos de software por medio de procesamiento de lenguaje natural"**

Después de haberlo leído, corregido e intercambiado información con el (la) alumno(a), y realizado los cambios que le fueron sugeridos, puede ser autorizada su impresión, a fin de que se inicien los trámites correspondientes para su defensa.

Sin otro particular por el momento, y en espera de que mis sugerencias sean tomadas en cuenta en beneficio del estudiante y la Institución, agradezco la atención prestada.

Atentamente,

Dra. Carolina Reta Castro

F31b Revisión: 04-Feb-2022

# CARTA DE LIBERACIÓN DEL REVISOR

Santiago de Querétaro, Qro., a 26 de enero de 2024.

Mtro. Geovany González Carlos
Gerencia de Posgrado
CIATEQ, A.C.

Por medio de la presente me dirijo a usted en calidad de Revisora del proyecto terminal del (la) alumno (a) Mtro. **Carlos Octavio Mendívil Vázquez**, cuyo título es:

"**Automatización de rastreabilidad de requerimientos de software por medio de procesamiento de lenguaje natural**"

Después de haberlo leído, corregido e intercambiado información con el (la) alumno(a), y realizado los cambios que le fueron sugeridos, puede ser autorizada su impresión, a fin de que se inicien los trámites correspondientes para su defensa.

Sin otro particular por el momento, y en espera de que mis sugerencias sean tomadas en cuenta en beneficio del estudiante y la Institución, agradezco la atención prestada.

Atentamente,

_____

Dra. Leonor Adriana Cárdenas Robledo

F31b Revisión: 04-Feb-2022

# AGRADECIMIENTOS

Agradezco a Dios por las personas que ha puesto en mi camino, particularmente por mis padres, hermana, amigos, maestros y mentores.

# RESUMEN

El desempeño de un modelo de lenguaje ajustado finamente mediante la transferencia de aprendizaje para automatizar el proceso de rastreabilidad de requerimientos depende del aprendizaje obtenido para representar la información textual de los requerimientos. Esto indica que para incrementar el desempeño en esta tarea es necesario estudiar técnicas de transferencia de aprendizaje enfocadas en la captura de la información semántica y sintáctica en los requerimientos. Por lo descrito anteriormente, se ha motivado a generar una propuesta de metodología de transferencia de aprendizaje de ajuste fino en dos etapas del modelo BERT que permite complementar la tarea de rastreabilidad de requerimientos de software. Esto consiste en entrenar en una primera etapa solamente las capas transformer de BERT que concentran la información semántica y continuando con una segunda etapa entrenando solamente las capas transformer de BERT que concentran la relación entre frases y la información sintáctica. El método propuesto se evalúo dentro de un marco de comparación contra los métodos de ajuste fino estándar, ajuste fino del 50 % del modelo y sondeo lineal entrenando solamente la capa de clasificación, obteniendo un incremento en la métrica de F2 de 6.38 %, 13.45 % y 65.72 %, respectivamente.

**Palabras clave:** BERT; Procesamiento de lenguaje natural (NLP); Rastreabilidad de requerimientos; Ingeniería de requerimientos; Tecnología de la automatización.

# ABSTRACT

Language model transfer learning performance on the downstream task to automate the traceability of software requirements depends on the ability to represent the requirements textual information. Then, to increase the model performance in this downstream task, it is required to develop transfer learning methods focused on capturing the semantic and syntactic information within the requirements. Therefore, this motivates to develop of a two-stage fine tuning transfer learning method proposal for BERT to collaborate with the software requirements traceability task. In the first stage only the BERT transformer layers that concentrate the semantic information are trained, next, in the second stage, only the BERT transformer layers that concentrate the syntactic information are trained. The proposed method was evaluated within a comparison framework against standard fine-tuning, fixing 50 % of model layers and classification layer linear probing, obtaining an F2 increment of 6.38 %, 13.45 % and 65.72 %, respectively.

**Keywords:** BERT; Natural Language Processing (NLP); Requirements traceability; Requirements engineering; Automation technology.

# ÍNDICE DE CONTENIDO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ABREVIATURAS y GLOSARIO

**AFE:** Ajuste fino estándar.

**AFF:** Ajuste fino fijo.

**BERT:** Por sus siglas en inglés "Bidirectional Encoder Representations from Transformers".

**Epochs:** Ciclos de entrenamiento.

**F1:** Puntaje empleado para representar el balance entre recuperación y precisión de manera equitativa.

**F2:** Puntaje empleado para representar el balance entre recuperación y precisión asignando mayor peso a recuperación.

**Linear probing:** Método de entrenamiento de sondeo lineal.

**LN:** Lenguaje natural.

**NLP:** Procesamiento de lenguaje natural, Natural Language Processing, por sus siglas en inglés.

**Pooler:** Capa de linealización de la salida de BERT hacia la capa de clasificación.

**VCE:** Validación cruzada externa.

**VCI:** Validación cruzada interna.

**Word embeddings:** Representación numérica de textos.

**WordNet:** Conjunto de grupos de sinónimos en inglés.

# 1. INTRODUCCIÓN

## 1.1. ANTECEDENTES

Los requerimientos constituyen el medio oficial para especificar todas las características y funcionamiento particular de un sistema (Tikayat Ray et al., 2023). Es decir, los requerimientos son la entrada formal para las distintas etapas dentro de la fase de desarrollo del sistema (Canedo y Mendes, 2020), como por ejemplo diseño, implementación, verificación, administración y documentación. Por lo tanto, la ingeniería de requerimientos representa una actividad que es crucial para obtener el resultado esperado de cada etapa, así como para detectar errores tempranamente en los mismos requerimientos (Tikayat Ray et al., 2023).

Por lo general, los requerimientos son representados en lenguaje natural (NL), ya que es una manera práctica y relativamente sencilla para redactarlos, pero en proyectos de sistemas complejos es común que durante el análisis de requerimientos se encuentren ambigüedades o huecos, provocando interpretaciones incorrectas (Lemazurier et al., 2017). Esto puede incrementar el tiempo invertido y la complejidad del mismo análisis (Sonbol et al., 2022a), lo cual ha promovido propuestas de automatización en la ingeniería de requerimientos, particularmente, la rastreabilidad de los requerimientos de software entre distintos niveles de abstracción ha sido objeto de estudio dentro de la industria de desarrollo de software para reducir el costo y esfuerzo requerido para crear y mantener la rastreabilidad entre requerimientos (Lin et al., 2022). Anteriormente se han empleado técnicas de procesamiento de lenguaje natural para automatizar la rastreabilidad entre requerimientos, pues la rastreabilidad entre requerimientos de software de distinto nivel de abstracción escritos en lenguaje natural (LN) se puede representar como un caso de clasificación binaria entre dos cadenas de texto (Lin et al., 2022). El propósito es evaluar cada una de las posibles combinaciones de pares entre ambos niveles de abstracción con el objetivo de determinar la condición de rastreabilidad entre cada uno de los pares para determinar si existe una rastreabilidad entre un par de requerimientos (clase positiva) o en su defecto, que no existe dicha rastreabilidad (clase negativa) (Lin et al., 2022; Tian et al., 2023).

Recientemente, se han hecho estudios relacionados al uso de modelos de lenguaje grandes previamente entrenados en una tarea de clasificación de texto general para llevar a cabo tareas relacionadas a la rastreabilidad de requerimientos. Estos estudios han sido principalmente motivados por la habilidad que presentan estos modelos para ser ajustados finamente por medio de técnicas de transferencia de aprendizaje (Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023).

Los modelos de lenguaje grande han superado otros métodos de procesamiento de lenguaje natural (NLP) en marcos de referencia establecidos para ejecutar tareas de clasificación de texto nuevas, particularmente, se ha elegido el modelo Bidirectional Encoder Representations from Transformers (BERT por sus siglas en inglés) por los resultados obtenidos en casos de entrenamiento fino para clasificación binaria (resultado positivo o negativo) de enunciados o secuencias de texto (Devlin et al., 2019). Adicionalmente, se han presentado propuestas de técnicas de transferencia de aprendizaje de entrenamiento en dos etapas en modelos de lenguaje grande de manera general (ValizadehAslani et al., 2022) y en el ámbito de rastreabilidad de requerimientos (Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023). De igual manera, hay estudios concernientes a la rastreabilidad de requerimientos donde se evalúa la técnica de entrenamiento en dos etapas empleando modelos de lenguaje pre-entrenados aplicando un remuestreo de las clases en alguna de las dos etapas para mejorar el desempeño del modelo en la detección de la clase minoritaria (Lin et al., 2021, 2022) o aplicando una técnica para dar más peso a la clase minoritaria en alguna de las dos etapas de entrenamiento (Tian et al., 2023).

Algunos de estos entrenamientos en dos etapas han hecho énfasis en la importancia de las últimas capas del modelo en el caso del entrenamiento de clases desbalanceadas (Fang et al., 2021), incluyendo modelos de lenguaje pre-entrenados como BERT (ValizadehAslani et al., 2022). Particularmente en Jawahar et al. (2019) se ha identificado que la información textual se concentra en distintas proporciones a lo largo de todas las capas transformer de BERT.

Anteriormente se han llevado a cabo estudios de la combinación de distintas capas de BERT en entrenamientos de una y dos etapas (Chen et al., 2023; Lee et al., 2019; Liu et al., 2021; ValizadehAslani et al., 2022; Yang et al., 2022). Sin embargo, aún no se ha hecho una propuesta de transferencia de aprendizaje que pretenda aprovechar la información semántica y sintáctica de las capas de BERT en la rastreabilidad de requerimientos. Por último, aunque las propuestas de técnicas en dos etapas mencionadas anteriormente han contribuido en el estudio de la transferencia de aprendizaje de un modelo de procesamiento de lenguaje natural (NLP por sus siglas en inglés) pre-entrenado desde una tarea general hacia una tarea particular para clasificar y determinar automáticamente condiciones de rastreabilidad entre requerimientos de software, aún es requerido incrementar el desempeño para esta tarea en el ámbito industrial (Canedo y Mendes, 2020; Lin et al., 2022; Rodriguez et al., 2023b, 2023a; Tian et al., 2023).

## 1.2. DEFINICIÓN DEL PROBLEMA

El problema por abordar se centra en el desempeño de un modelo de lenguaje grande ajustado finamente para la tarea específica de rastreabilidad de requerimientos en el contexto del desarrollo de software.

El desempeño de un modelo de lenguaje ajustado finamente en la tarea de rastreabilidad de requerimientos depende del aprendizaje obtenido para representar la información textual de los requerimientos (Canedo y Mendes, 2020; Lin et al., 2022; Rodriguez et al., 2023b, 2023a; Tian et al., 2023). Esto indica que para incrementar el desempeño en esta tarea es necesario estudiar técnicas de transferencia de aprendizaje enfocadas en la captura de la información semántica y sintáctica en los requerimientos (Sonbol et al., 2022b).

Por lo descrito anteriormente, se ha motivado a proponer un método de transferencia de aprendizaje que aporte en el desempeño del modelo para automatizar la rastreabilidad de requerimientos de software.

## 1.3. JUSTIFICACIÓN

La rastreabilidad entre requerimientos de software permite relacionar las etapas del ciclo de vida de desarrollo, desde el origen del proyecto hasta su comercialización y mantenimiento (Lin et al., 2022). Además es requerida por entidades de certificación y regulatorias de desarrollo de software (Tikayat Ray et al., 2023).

Recientemente se ha demostrado que la transferencia de aprendizaje con una etapa intermedia entre el pre-entrenamiento y el ajuste final del modelo de lenguaje han permitido obtener un incremento en el desempeño del modelo final tanto en el área de NLP en general como en el ámbito particular de la rastreabilidad de requerimientos (Lin et al., 2021, 2022; Rodriguez et al., 2023b; ValizadehAslani et al., 2022). Por lo tanto, el presente trabajo busca extender el estado del arte proponiendo un nuevo método de transferencia de aprendizaje de ajuste fino en dos etapas del modelo de lenguaje natural BERT base, que se describe a alto nivel a continuación.

## 1.4. PROPUESTA DE SOLUCIÓN

La presente tesis propone un método de transferencia de ajuste fino en dos etapas que permite complementar la tarea de rastreabilidad de requerimientos de software. Esto se realiza entrenando en una primera etapa solamente las capas transformer de BERT que concentran la información semántica y continuando con una segunda etapa entrenando solamente las capas transformer de BERT que concentran la relación entre frases y la información sintáctica.

La presente propuesta de solución es motivada por las siguientes razones:

- Es posible que una mejor representación sintáctica y semántica del requerimiento mejore la tarea de rastreabilidad de requerimientos (Sonbol et al., 2022b), por lo tanto, se justifica enfocar el aprendizaje en las capas que concentran esta información durante el ajuste fino.
- El entrenamiento de todas las capas transformer puede llevar al olvido catastrófico del aprendizaje afectando el desempeño del modelo. Este

efecto se ve disminuido cuando se reduce la cantidad de capas transformer durante la transferencia de aprendizaje (Kumar et al., 2022).

- El agrupamiento de capas transformer con la misma función durante la transferencia de aprendizaje puede incrementar el desempeño del modelo (Jawahar et al., 2019; Lee et al., 2019; Liu et al., 2021).
- El entrenamiento en dos etapas puede ayudar al modelo de lenguaje y a la capa de clasificación a incrementar el desempeño final en la tarea objetivo (Lin et al., 2021, 2022; Tian et al., 2023; ValizadehAslani et al., 2022).

## 1.5. OBJETIVOS

### 1.5.1. Objetivo general

Estudiar el problema de rastreabilidad de requerimientos de software por medio de procesamiento de lenguaje natural a partir de la propuesta de un método de transferencia de aprendizaje en dos etapas del modelo pre-entrenado de BERT base que considere la distribución de la información semántica y sintáctica entre las capas transformer de BERT.

### 1.5.2. Objetivos específicos

1. Investigar el estado del arte con respecto a las metodologías de ajuste fino en el área de rastreabilidad de requerimientos de software.
2. Analizar las funciones de las capas de BERT y las técnicas de transferencia de aprendizaje que consideren dichas funciones durante el entrenamiento.
3. Proponer un método de transferencia de aprendizaje de ajuste fino en dos etapas para la rastreabilidad de requerimientos de software considerando la concentración de la información semántica y sintáctica entre las capas de BERT.
4. Desarrollar un marco de comparación entre el método propuesto contra tres métodos de referencia.

## 1.6. HIPÓTESIS

A partir del método de transferencia de aprendizaje de ajuste fino en dos etapas propuesto en el modelo base de BERT, se puede obtener un incremento del 1 % en F2 (puntaje empleado para representar el balance entre recuperación y precisión) en la tarea de rastreabilidad de requerimientos de software en una misma base de datos en comparación con las siguientes tres técnicas de transferencia de aprendizaje en el modelo base de BERT:

- Ajuste fino estándar.
- Ajuste fino fijando el 50 % de BERT a partir de la sexta capa transformer.
- Ajuste fino por medio de sondeo lineal (Linear probing).

# 2. MARCO TEÓRICO

## 2.1. GESTIÓN DE REQUERIMIENTOS DE SOFTWARE POR MEDIO DE NLP

El análisis de requerimientos es una tarea científicamente relevante debido a que es computacionalmente complejo entender el significado correcto de una palabra de acuerdo con su contexto (Croft et al., 2013), y la importancia del correcto manejo de requerimientos en la cadena de valor y el ciclo de desarrollo de software. Aún para una persona, el reconocimiento del significado de una palabra de acuerdo al contexto no es una actividad simple, por ejemplo, Matsuoka y Lepage (2011) desarrollaron un método que permite revisar requerimientos para detectar términos ambiguos mediante el uso de WordNet complementado con metodologías de análisis de frecuencia y significado técnico. WordNet (Miller, 1994) es un conjunto de grupos de sinónimos en inglés, es decir, por cada significado diferente que posee una palabra ésta forma parte de un grupo distinto, por lo cual, es comúnmente empleada en estudios de análisis semántico de texto con base en ontologías y taxonomías. En Sagala et al. (2018) se presenta una visión comparativa de la efectividad de WordNet para realizar análisis semántico contra otras opciones, y aunque son resultados favorables hacia WordNet con respecto a otras metodologías similares, también se observa que los resultados pueden ser afectados por la taxonomía empleada para calcular la similitud semántica entre términos.

El uso de ontologías en el manejo de requerimientos es motivado por el hecho de que la efectividad de las ontologías como forma de clasificación de conceptos. Por ejemplo en Jayatilleke y Lai (2013) se propone una metodología para el control de cambios de requerimientos basada en una ontología y terminología específica. El uso de ontologías para análisis de texto ha obtenido resultados efectivos, sin embargo, la mayoría se apoyan en métodos basados en reglas taxonómicas hechas a mano para determinar la similitud semántica. Esta estrategia encuentra una importante limitante para el área industrial, ya que para cada proyecto tendría que crear para sí un conjunto de reglas semánticas y taxonómicas.

Otra aplicación de NLP es la revisión de requerimientos con respecto a plantillas preestablecidas debido a que ésta puede ser una tarea tediosa y repetitiva. En Arora et al. (2015) se propone un método automático de revisión de requerimientos contra una plantilla predefinida previamente en un patrón semántico mediante NLP, obteniendo resultados aceptables. Pero, este método, está estrechamente ligado a una definición previa de los requerimientos con respecto a la plantilla que se va a revisar, lo cual, es una limitación ya identificada en Jayatilleke y Lai (2013).

Además, en otros trabajos publicados (Rago et al., 2016) se ha identificado la necesidad de apoyar al personal que analiza los requerimientos para encontrar consideraciones de diseño implícitas, por lo cual han propuesto un herramienta basada en NLP que permite a la persona responsable detectar estas posibles consideraciones. En Rago et al. (2016) se hace énfasis en que la precisión de la herramienta, aunque es similar a otras opciones, es menor a la realizada por una persona. Como alternativa, Wang (2016) propone un método automático para analizar requerimientos de software que combina WordNet y técnicas de aprendizaje automático (Machine Learning) para extraer su contenido semántico.

Con respecto al análisis del impacto de los cambios en un requerimiento, en Arora et al. (2015) se propone un método basado en NLP que se apoya en métodos cuantitativos para determinar el grado de impacto que puede sufrir un requerimiento bajo ciertas condiciones de cambio y aunque se presentan resultados positivos aún se requiere de más estudios de factibilidad de uso en campo industrial. Priya y Anupriya (2013) presentan una propuesta que complementa WordNet con lógica difusa, pero es aplicada a enunciados simples, los cuales representan contexto diferente al de los requerimientos de software, y en consecuencia es una técnica que tendría importantes limitaciones al buscar ser aplicada en el contexto de requerimientos de software.

En Y. Li et al. (2019) se propone una arquitectura para generar estructuras jerárquicas de repositorios de requerimientos con el objetivo de reutilizarlos en futuros productos, pero no se aborda la rastreabilidad entre requerimientos. En Zhao

et al. (2017) se emplean Word Embeddings en combinación con técnicas de aprendizaje automático, específicamente se usa Learning to Rank, obteniendo resultados favorables en la rastreabilidad de artefactos de software, pero en proyectos de dominio no industrial.

Mahmoud y Williams (2016) proponen un método para rastrear y clasificar requerimientos de software pero se enfocan solamente en requerimientos no funcionales, limitando el alcance de su técnica al conjunto de requerimientos funcionales, el cual tiene amplia relevancia en el contexto industrial. En el trabajo de Lucassen et al. (2016) se desarrolló una herramienta que ejecuta automáticamente el análisis semántico de requerimientos mediante WordNet, pero su alcance se limita a encontrar requerimientos duplicados.

En Bella et al. (2018) se propone un método basado en aprendizaje automático semi-supervisado, lo que permite emplear datos no etiquetados para el entrenamiento de un modelo, y aunque obtuvieron resultados que demuestran que es factible utilizar este método para la rastreabilidad de requerimientos, todavía no se ha experimentado en casos industriales. Por otra parte, en McZara et al. (2015) emplean análisis de similitud léxica para identificar pares de requerimientos similares con el objetivo de priorizar requerimientos pero no se aborda la rastreabilidad de requerimientos. En Misra (2016) se plantea el objetivo de encontrar alias semánticos que se refieren en a la misma entidad dentro del mismo texto, mediante técnicas de NLP y aprendizaje automático. Por último, en Wu et al. (2017) se combinan técnicas de aprendizaje profundo y aprendizaje automático para obtener la similitud semántica entre enunciados, pero no aplica en caso de textos cortos o párrafos.

En Unterkalmsteiner et al. (2016) se evalúa la factibilidad de emplear modelos de NLP para evaluar la similitud semántica entre artefactos de software, pero su alcance es con respecto a la relación entre casos de prueba y código fuente. Por otra parte, en J. Guo et al. (2017) se describe un método que combina Word Embeddings con redes neuronales recursivas (RNN por sus siglas en inglés),

obteniendo resultados superiores comparados con otros modelos de NLP, aplicados en un caso práctico industrial, pero con la limitante de que es una solución para un dominio específico por lo cual carece de generalidad.

Consecuentemente, en el mismo rubro de Word Embeddings, los modelos de lenguaje han sido estudiados recientemente en la ingeniería de requerimientos en general, a raíz del éxito obtenido por BERT en tareas de clasificación de texto ajustando finamente el modelo por medio de transferencia de aprendizaje (Sonbol et al., 2022b; Zheng et al., 2023). Lo cual, marcó la ventaja principal de estos modelos, por ejemplo BERT, contra técnicas de Word Embeddings previas (como Word2Vec) (Shen y Liu, 2021), pues anteriormente solo se había logrado una representación del contexto de manera unidireccional (Devlin et al., 2019).

Particularmente, en el área de rastreabilidad de requerimientos se han presentado estudios aplicando técnicas de transferencia de aprendizaje en modelos de lenguaje previamente entrenados que han superado los métodos previos de NLP (Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023).

Inicialmente Lin et al. (2022) demostraron que emplear una técnica de transferencia implementando un paso de entrenamiento intermedio de BERT en una tarea similar antes de ejecutar el ajuste fino en la tarea de clasificación binaria de rastreabilidad de requerimientos, se puede obtener un incremento de manera general dentro de un marco de referencia común en comparación contra los métodos tradicionales de NLP y método de ajuste fino estándar de BERT. Para llevar a cabo este entrenamiento intermedio Lin et al. (2022) primero minaron una base de datos para la tarea intermedia entre un conjunto de proyectos de software disponibles públicamente, obteniendo un total de 110GB de datos.

Después, Lin et al. (2022) ejecutaron el ajuste fino de BERT para la tarea intermedia que consistió en la identificación de rastreabilidad entre la descripción de un problema y la propuesta de solución correspondiente a ese problema.

Posteriormente, se hizo el ajuste fino de BERT de manera estándar en la base de datos objetivo en la tarea de rastreabilidad de requerimientos.

Rodriguez et al. (2023b) siguieron los pasos propuestos por Lin et al. (2022) de igual manera en un marco de comparación obteniendo un incremento en desempeño sobre el método de ajuste fino estándar de BERT. Esto sustenta los resultados obtenidos por Lin et al. (2022).

Adicionalmente, inspirados por los resultados de Lin et al. (2022), en el trabajo de Tian et al. (2023), se ejecutó un segundo entrenamiento para adaptar BERT al dominio del proyecto bajo estudio. En este caso, primero se recopiló una base de datos intermedia con la descripción de los requerimientos bajo estudio y en segundo lugar, BERT fue entrenado de nuevo en sus tareas originales (entrenamiento enmascarado y predicción de siguiente secuencia) empleando la base de datos intermedia. Sin embargo, Tian et al. (2023) no llevaron a cabo una etapa de ajuste fino final, en su lugar, combinaron los Word Embeddings de BERT con la información histórica del requerimiento recopilada durante el desarrollo del proyecto de software. Esto significa que extrajeron la información sobre el autor del requerimiento, historial de cambios en el requerimiento, comentarios relacionados al requerimiento, entre otros. Como resultado, Tian et al. (2023) demostraron que enriquecer la representación textual de BERT puede incrementar el desempeño del modelo en la rastreabilidad de requerimientos. Cabe aclarar que aunque no implementaron un ajuste fino, concluyeron que un ajuste fino después de adaptar a BERT al dominio del proyecto podría incrementar el desempeño del modelo.

Con base en el análisis bibliográfico anterior se identifican los siguientes aportes:

- Las representaciones textuales de requerimientos de software generadas por medio de modelos de lenguaje permiten obtener un mayor desempeño que métodos de NLP previos a BERT en la tarea de rastreabilidad de requerimientos por su característica bidireccional (Lin et al., 2022; Rodriguez et al., 2023b; Sonbol et al., 2022b; Tian et al., 2023; Zheng et al., 2023).

- Una fase de entrenamiento con el objetivo de adaptar el modelo de lenguaje al dominio del proyecto puede incrementar el desempeño del modelo en la tarea de rastreabilidad de requerimientos (Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023).
- Ajustar finamente el modelo después de una etapa intermedia de entrenamiento en una base de datos diferente a la base de datos objetivo puede incrementar el desempeño final del modelo en la rastreabilidad de requerimientos (Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023).

De la misma manera, se identifica que en los casos analizados de los modelos de lenguaje para la tarea de rastreabilidad de requerimientos existen las siguientes oportunidades de mejora:

- Se tiene que llevar a cabo la tarea adicional de recopilar o generar una segunda base de datos intermedia o auxiliar para llevar a cabo una fase de entrenamiento para adaptar al modelo al dominio del proyecto.
- Además, es necesario identificar una tarea para la fase de adaptación que permita al modelo adaptarse al dominio del proyecto.
- Finalmente, es notable que ningún trabajo previo se ha propuesto hacer énfasis en algún grupo de capas transformer en particular, ni durante el entrenamiento intermedio ni en el ajuste final, con el objetivo de incrementar el desempeño del modelo.

Estas oportunidades de mejora identificadas en la bibliografía analizada han motivado a desarrollar el método propuesto en esta tesis. En la Tabla 1 se comparan los trabajos actuales contra el método propuesto en esta tesis según las características de modelo de NLP, las etapas de transferencia de aprendizaje, métodos de transferencia de aprendizaje y tipo de fuente de la base de datos en cada uno de los trabajos analizados.

Tabla 1. Comparativa de literatura contra el método propuesto

| Característica / Referencia en la literatura | | Lin et al., 2022 | Rodriguez et al., 2023b | Tian et al., 2023 | Método propuesto |
|---|---|---|---|---|---|
| Modelo | | BERT | BERT | BERT | BERT |
| Etapas de transferencia de aprendizaje | Entrenamiento en una etapa | SI | SI | SI | NO |
| | Entrenamiento en dos etapas | SI | SI | NO | SI |
| Método de transferencia de aprendizaje en primera etapa | Entrenamiento de todo el modelo | SI | SI | SI | NO |
| | Entrenamiento selectivo de las capas de BERT | NO | NO | NO | SI |
| Base de datos en primera etapa | Base de datos auxiliar dentro del dominio de la tarea objetivo | SI | SI | SI | NO |
| | Base de datos de la tarea objetivo | NO | NO | NO | SI |
| Tarea de entrenamiento en primera etapa | Diferente a la tarea objetivo | NO | NO | SI | NO |
| | Similar a la tarea objetivo | SI | SI | NO | NO |
| | Idéntica a la tarea objetivo | NO | NO | NO | SI |
| Método de transferencia de aprendizaje en segunda etapa | Entrenamiento de todo el modelo | SI | SI | - | NO |
| | Entrenamiento selectivo de las capas | NO | NO | - | SI |
| Base de datos en segunda etapa | Base de datos auxiliar dentro del dominio de la tarea objetivo | NO | NO | - | NO |
| | Base de datos de la tarea objetivo | SI | SI | - | SI |
| Tarea de entrenamiento en segunda etapa | Diferente a la tarea objetivo | NO | NO | - | NO |
| | Similar a la tarea objetivo | NO | NO | - | NO |
| | Idéntica a la tarea objetivo | SI | SI | - | SI |
| Método de generación de base de datos auxiliar | Recopilación manual de documentos del proyecto | NO | NO | SI | - |
| | Minado entre descripciones de problemas y propuestas de solución | SI | SI | NO | - |
| Fuente de bases de datos | Fuente abierta | SI | SI | SI | SI |
| | Colaborador industrial | SI | NO | NO | NO |

Elaboración propia

## 2.2. CONCEPTOS BÁSICOS DE CLASIFICACIÓN

En esta sección se describen los conceptos básicos de clasificación sobre los cuales se ha sustentado el trabajo desarrollado en esta tesis.

### 2.2.1. Clasificación binaria

La clasificación binaria se caracteriza por recibir un elemento o un grupo de elementos como entrada, para posteriormente determinar la probabilidad de que dicha entrada pertenezca a una de dos clasificaciones posibles(J. J. Li y Tong, 2020). En la Figura 1 se puede observar un ejemplo de un clasificador binario que tiene como salida el porcentaje de probabilidad de que la entrada pertenezca a una de las dos posibles clases, en este caso, a la Clase 1 o a la Clase 0:


Figura 1. Clasificador binario
Elaboración propia

### 2.2.2. Matriz de confusión

La matriz de confusión permite observar de manera gráfica el desempeño de un clasificador binario con base a un resultado positivo o negativo, es decir, sí pertenece o no a una de las dos clases posibles. Como se muestra en la **Error! Reference source not found.**, el total de verdaderos positivos (TP por sus siglas en inglés) es el número de predicciones correctas como positivas y el total de falsos positivos (FP por sus siglas en inglés) es el número de predicciones erróneamente asignadas como positivas, de tal manera que la suma de TP y FP es el número Total de predicciones positivas que el clasificador hizo.

Mientras que el total de verdaderos negativos (TN por sus siglas en inglés) es el número de predicciones correctas como negativas y el total de falsos negativos (FN por sus siglas en inglés) es el número de predicciones erróneamente asignadas como negativas, por lo tanto, la suma de TN y FN es la cantidad total de predicciones negativas realizadas por el clasificador (Chawla, 2005).

Tabla 2. Matriz de confusión con base a la verdad fundamental

| | | **Resultados de predicción del modelo** | |
|---|---|---|---|
| | | **Negativos** | **Positivos** |
| **Valores actuales** | **Negativos** | Total verdaderos negativos (TN) | Total falsos positivos (FP) |
| | **Positivos** | Total falsos negativos (FN) | Total verdaderos positivos (TP) |

Elaboración propia

### 2.2.3. Recuperación, Precisión, F1 y F2

Recuperación es la capacidad del modelo para identificar todas las muestras relevantes y precisión es la habilidad del modelo para identificar sólo instancias relevantes. Por otra parte, F1 es un puntaje que busca representar el balance entre recuperación y precisión de manera equitativa (Chawla, 2005), mientras que F2 le da el doble de importancia a la recuperación, pues le da más valor a la recuperación sin dejar de tomar en cuenta el valor de la precisión.

Visto de otra manera, recuperación es igual al total de positivos verdaderos dividido entre la sumatoria de los positivos verdaderos y los falsos negativos. Por otro lado, precisión es igual al total de positivos verdaderos dividido entre la sumatoria de los positivos verdaderos y los falsos positivos, tal como se muestra en las siguientes ecuaciones (Chawla, 2005):

$$Recuperación = TP * (TP + FN)$$

Ecuación 1. Cálculo de recuperación

$$Precisión = TP * (TP + FP)$$

Ecuación 2. Cálculo de precisión

Por último, el puntaje de F1 y F2 se pueden calcular de la siguiente manera:

$$F1 = 2 * (\frac{Precisión * Recuperación}{Precisión + Recuperación})$$

Ecuación 3. Cálculo de F1

$$F2 = 5 * (\frac{Precisión * Recuperación}{4 * (Precisión + Recuperación)})$$

Ecuación 4. Cálculo de F2

La ventaja de usar F1 y F2 es que ofrecen una métrica de evaluación de un clasificador binario de una manera objetiva (Chawla, 2005).

### 2.2.4. Criterios de evaluación de clasificadores binarios

Un clasificador binario puede presentar variabilidad en su desempeño debido al orden y la distribución de las muestras durante el entrenamiento, por lo que es común emplear métodos de validación cruzada dentro de un marco de comparación en una tarea en común (J. J. Li y Tong, 2020), pues esto permite analizar objetivamente el desempeño del modelo, evitando caer en un caso aislado donde el orden y la distribución de las muestras durante el entrenamiento fue de particular beneficio para el modelo evaluado. A continuación se describen los métodos de validación cruzada y de marco de referencia de comparación que se emplearon en esta tesis:

- **Validación cruzada:** Consiste en dividir la base de datos en un numero k de subconjuntos no superpuestos, dejando k-1 subconjuntos para el entrenamiento y 1 subconjunto para la prueba del modelo. Este proceso se repite k veces rotando el grupo de prueba y los grupos de entrenamiento reiniciando el entrenamiento en cada ronda, es decir, cada subconjunto es usado al menos una vez como grupo de prueba. Al final de todas las rondas, el resultado reportado es el promedio de todas las rondas (J. J. Li y Tong, 2020).
- **Doble validación cruzada:** Básicamente consiste en ejecutar dos validaciones cruzadas, una externa para reportar resultados y una interna para optimizar los hiperparámetros del modelo. Se divide la base de datos en k subconjuntos por medio de una validación cruzada (externa), después se prosigue a optimizar los hiperparámetros del modelo aplicando una validación cruzada (interna) a los k-1 subconjuntos del grupo de entrenamiento de la validación cruzada externa, es decir, se obtienen n

subconjuntos a partir de los grupos de entrenamiento k-1 para tener n-1 subconjuntos de entrenamiento interno y 1 subconjunto para validación. Los hiperparámetros se optimizan en la n rondas de la validación cruzada interna según los resultados en el grupo de validación acorde a la correspondiente ronda de la validación cruzada externa (May et al., 2022). Esto permite optimizar los hiperparámetros manteniendo aislado el subconjunto de prueba durante todo el proceso (Wainer y Cawley, 2018). El resultado final del modelo es el promedio de todas las rondas de la validación cruzada externa. Particularmente, en esta tesis la optimización de hiperparámetros se realizó mediante una búsqueda exhaustiva.

- **Marco de referencia de comparación:** Consiste en comparar objetivamente la métrica del modelo con otros modelos, se elabora un marco de comparación en una tarea en común, la cual consiste en que todos los modelos son entrenados empleando las mismas muestras y posteriormente son evaluados con los mismos grupos de pruebas mediante los resultados obtenidos en la misma métrica (J. J. Li y Tong, 2020). A partir de este punto será conocido como marco de comparación.

### 2.2.5. Técnicas para manejo de base de datos desbalanceadas en clasificadores binarios

Existen distintos métodos y técnicas para balancear una base de datos, (Last et al., 2017; Madabushi et al., 2020). En seguida se describen los utilizados en esta tesis:

- **Remuestreo:** Técnica de remuestreo aleatorio dinámico, la cual consiste en reducir la cantidad de muestras de clase minoritaria o en aumentar la cantidad de muestras de la clase mayoritaria hasta obtener un balance en la distribución de las clases. Particularmente en el área de rastreabilidad de requerimientos se ha ejecutado la técnica de remuestreo a nivel de paquetes, es decir, aumentando la cantidad de muestras de la clase minoritaria en un conjunto de paquetes que son balanceados de manera dinámica durante el entrenamiento (Guo et al., 2018).
- **Reponderación de peso de las clases:** Se le asigna un peso en particular a cada clase en la función de costo durante el entrenamiento, dándole un

mayor peso a la clase minoritaria. Esfuerzos anteriores han demostrado que se puede asignar un peso proporcional a la frecuencia de la clase en la rastreabilidad de requerimientos durante la transferencia de aprendizaje (Tian et al., 2023).

### 2.2.6. Red neuronal prealimentada para clasificación

Una red neuronal prealimentada es un modelo diseñado con el objetivo de aproximar una función de múltiples entradas y salidas, por ejemplo para un clasificador la red neuronal aproxima una relación entre una entrada X hacia una categoría Y (Goodfellow et al., 2016). Básicamente, está compuesta por una capa de entrada y una capa de salida, además, usualmente también incorporan una capa oculta. Las capas de entrada y de salida permiten que la red se conecte con el mundo exterior, mientras que la capa oculta extrae patrones característicos de la información de entrada (Haykin, 2001).

En la Figura 2 se muestra una red neuronal prealimentada de una capa de entrada de 2 nodos, una capa de salida de 2 nodos y una capa oculta de 3 nodos:

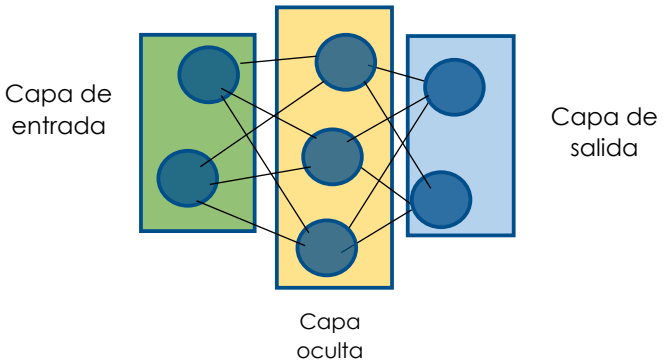



Figura 2. Ejemplo de arquitectura de red neuronal prealimentada
Elaboración propia

Como se puede observar en la Figura 2 cada nodo o neurona de cada capa está conectada con todos los nodos de la siguiente capa y así sucesivamente por cada capa que conforme a la red. La información fluye de izquierda a derecha, siendo

transformada en cada nodo por un conjunto de parámetros (como "weigths" y "bias") y por una función de activación previamente configurada. La función de unidad lineal rectificada (rectified linear unit en inglés o ReLU abreviado) y la función tangente son usadas como función de activación en redes neuronales prealimentadas (Goodfellow et al., 2016).

Para que la red neuronal sea capaz de aproximar una relación entre la entrada X y la salida Y, como en el caso de un clasificador, se lleva a cabo un proceso de entrenamiento en el cual los parámetros de la red son adaptados por un proceso continuo de aprendizaje. Este proceso es conocido como método de Backpropagation, el cual consiste en ajustar los parámetros de la red con el objetivo de minimizar el error en la predicción obtenida por el modelo de las muestras (Goodfellow et al., 2016). Si las muestras fueron previamente etiquetadas en la base de datos antes se le conoce como entrenamiento supervisado, en caso contrario es un entrenamiento no supervisado.

El error es calculado mediante una función conocida como "función de costo" y el cambio o ajuste en los parámetros es determinado mediante un algoritmo de optimización u optimizador, los cuales son seleccionados dependiendo de la aplicación y el tipo de información analizada por el modelo (Goodfellow et al., 2016). En el presente documento se empleó arbitrariamente la función de entropía cruzada (Cross entropy loss function en inglés) para calcular el error entre la predicción del modelo y el valor real esperado (Goodfellow et al., 2016). Con respecto al optimizador, se empleó Adam, siguiendo la recomendación de ajuste fino descrita en (Devlin et al., 2019). Adam (Kingma y Ba, 2017; Reddi et al., 2019) es una variante de gradiente descendente aleatorio, que tiene como objetivo determinar la dirección en la que la función debe avanzar para encontrar el mínimo o el máximo en cada iteración, en este caso se busca minimizar el error (Goodfellow et al., 2016). Cabe aclarar que la magnitud de los pasos en dicho avance se determina mediante un parámetro conocido como tasa de aprendizaje (Learning rate, en inglés) el cual se adapta según la distribución de los parámetros (Goodfellow et al., 2016; Haykin, 2004).

En el escenario donde el objetivo es la clasificación, como es el caso de este documento, es necesario añadir una función de salida que permita normalizar el valor de salida, de tal manera que se obtenga la probabilidad de que la muestra pertenezca a cada una de las clases, ya que la salida de la red no representa un porcentaje de probabilidad por sí misma(Goodfellow et al., 2016). La función Softmax es una opción para normalizar la salida de tal manera que la suma de las salidas de la red sea igual a uno (Goodfellow et al., 2016). Esto permite determinar la clasificación a la cual pertenece la entrada, en la Figura 3 se muestra un ejemplo de un clasificador binario basado en la red neuronal descrita en la Figura 2.

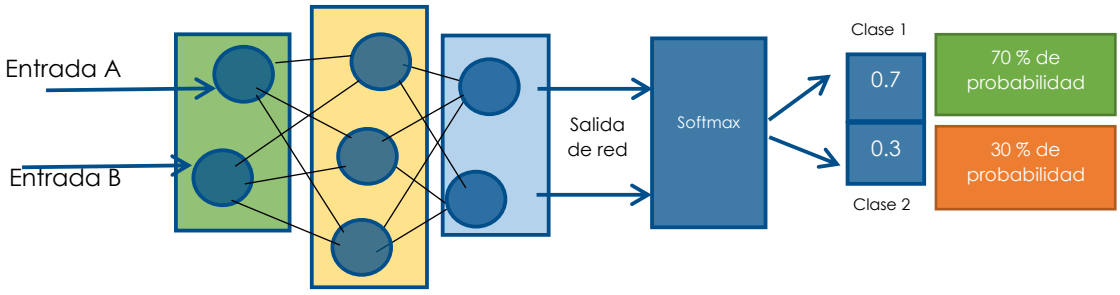



Figura 3. Clasificador binario basado en una red neuronal prealimentada
Elaboración propia

En el caso estudiado en esta tesis se añade una capa de clasificación binaria a un modelo de lenguaje previamente entrenado y el entrenamiento se va a ejecutar por medio de transferencia de aprendizaje. Particularmente, en NLP existen varias propuestas de modelos de aprendizaje profundo pre-entrenados, los cuales se describen a continuación.

## 2.3. MODELOS DE LENGUAJE PRE-ENTRENADOS AJUSTABLES FINAMENTE

Son modelos de lenguaje que fueron entrenados previamente con cuerpos de texto en el rango de millones de palabras, como por ejemplo BERT (Devlin et al., 2019) y OpenAI GPT (Radford y Narasimhan, 2018), con el objetivo de ser reutilizados en tareas específicas por medio del ajuste fino de sus parámetros.

BERT y OpenAI GPT se basan en la arquitectura Transformador (Transformer en inglés) (Devlin et al., 2019; Radford y Narasimhan, 2018), la cual utiliza múltiples

bloques de atención, cada bloque transforma mediante capas lineales cada secuencia para aplicarle pesos de atención, la cual le permite identificar la relevancia de las palabras en una secuencia (Vaswani et al., 2023).

Originalmente la arquitectura transformer fue propuesta con el propósito de traducción de lenguaje, por lo que está compuesta por un codificador (encoder) y por un decodificador (decoder), ambos implementan el mecanismo de atención para obtener una representación numérica de la secuencia de texto (Word Embeddings) (Vaswani et al., 2023):

- **Encoder:** Provee la representación semántica del texto de entrada, analizando simultáneamente la relación que cada palabra tiene con respecto a las demás por medio de los mecanismos de atención, como se muestra en la Figura 4 (Vaswani et al., 2023).
- **Decoder:** Tiene el objetivo de generar el texto de salida con base en la representación captura por el encoder. La principal diferencia con el encoder, es que el decoder implementa los mecanismos de atención solo en las palabras previas, es decir, tiene una dirección de izquierda a derecha (Devlin et al., 2019; Radford y Narasimhan, 2018).

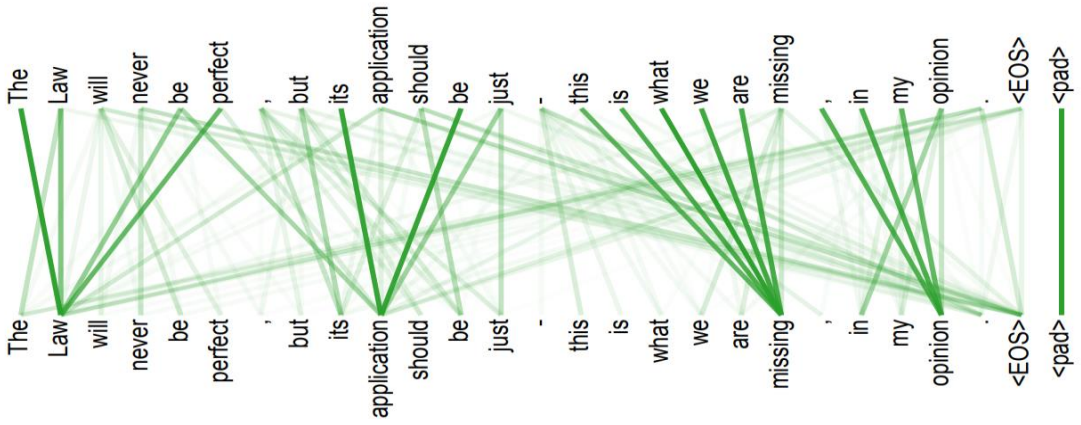



Figura 4. Representación visual del mecanismo de atención
(Vaswani et al., 2023)

Por lo tanto, los modelos de lenguaje con arquitectura transformer pueden ser capaces de capturar patrones y representaciones textuales dependiendo de la

estructura de bloques de encoders y decoders transformers que implementa el modelo (Devlin et al., 2019; Radford y Narasimhan, 2018; Vaswani et al., 2023):

- **Pila de Encoder-Decoders:** Es la estructura tradicional de la arquitectura transformer, y está compuesta por una pila de encoders que primero captura la representación semántica de la entrada y la pasa una pila de decoders con el objetivo de generar la secuencia de salida con relación a la representación semántica de la entrada.
- **Pila de Decoders:** Tienen un objetivo generativo, por lo tanto, representan un modelo unidireccional que analiza la secuencia de izquierda a derecha para generar la salida del modelo. Tal es el caso de GPT (Radford y Narasimhan, 2018).
- **Pila de Encoders**: Su función es obtener la relación que cada palabra tiene con todas las demás, no solo con las palabras previas, pues su objetivo es obtener una representación semántica de cada palabra de manera bidireccional a la salida del modelo en forma de Word Embeddings (Devlin et al., 2019).

Adicionalmente, el desempeño de la arquitectura transformer en la tarea final depende del proceso de entrenamiento previo y del tamaño del modelo, es decir, la cantidad de capas transfomer, del número de cabezas de atención en cada capa transformer y de la profundidad de los Word Embeddings (Devlin et al., 2019; Radford y Narasimhan, 2018; Vaswani et al., 2023). Lo cual, ha resultado en una serie de variantes tanto de GPT (Brown et al., 2020; Radford et al., 2019; Radford y Narasimhan, 2018) como de BERT (Devlin et al., 2019; Feng et al., 2020; D. Guo et al., 2020; Liu et al., 2019; Sanh et al., 2019).

Debido a los resultados obtenidos por BERT en las tareas de clasificación y representación de texto, se han propuesto variantes que han mejorado su desempeño de manera general y su eficiencia en el consumo de recursos (Liu et al., 2019; Sanh et al., 2019). Por ejemplo, RoBERTa es una versión que ha optimizado el proceso de preentrenamiento de BERT (Liu et al., 2019), mientras que DistilBERT es

una variante de menor tamaño que busca ser más eficiente en tiempo de ejecución y consumo de recursos en tareas generales (Sanh et al., 2019).

Así mismo, también hay variantes en el entrenamiento previo de BERT con el objetivo de especializarse en tareas específicas dentro de un dominio particular (Ajagbe y Zhao, 2022; Feng et al., 2020; D. Guo et al., 2020; X. Li et al., 2022), por ejemplo, algunas variantes notables especializadas en la ingeniería de software son CodeBERT y GraphCodeBERT, ambas se especializa en aplicaciones relacionando lenguaje natural y lenguaje de programación como la búsqueda de código (Feng et al., 2020; D. Guo et al., 2020).

En esta tesis se empleó el modelo BERT original para llevar a cabo la transferencia de aprendizaje, porque el objetivo es obtener una representación semántica de manera bidireccional de los requerimientos de software y porque la versión original ha sido recomendada por estudios previos concernientes a la rastreabilidad de requerimientos de software(Lin et al., 2022; Rodriguez et al., 2023b; Tian et al., 2023).

Los autores originales de BERT generaron dos modelos: "BASE" y "LARGE" (Devlin et al., 2019), por lo tanto, en el caso particular del presente documento, se empleó BERT "BASE" al cual se harán referencias futuras solo como BERT.

En la siguiente sección se describe en detalle el modelo BERT, es decir, su arquitectura, así como procesos de entrenamiento previo y de ajuste fino estándar.

### 2.3.1. Modelo BERT

BERT fue diseñado con el propósito de proveer un modelo pre-entrenado para ser ajustado finamente para tareas de NLP específicas, por ejemplo, analizar un grupo de preguntas con el objetivo de determinar sí son semánticamente equivalentes.

BERT ofrece la posibilidad de procesar como entrada un enunciado o dos enunciados concatenados en una sola secuencia dependiendo del uso requerido. Para interpretar una secuencia de texto BERT requiere que el texto a analizar sea

separado en una cadena de palabras y al mismo tiempo se incorporen una serie de identificadores de la siguiente manera (Devlin et al., 2019):

- El identificador [CLS] Se agrega al inicio de cada secuencia de texto.
- El identificador [SEP] Se utiliza para separar enunciados en el caso de que la secuencia contenga dos enunciados.
- Identificador correspondiente al vocabulario predefinido para BERT.

En el caso particular del presente trabajo, se procesa una secuencia de dos enunciados, como se muestra en la Figura 5:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| [CLS] | contenidd | en | texto | A | [SEP] | contenidd | en | texto | B | [SEP] |

Figura 5. Secuencia de entrada para BERT
Elaboración propia

La secuencia representada en la Figura 5 es empleada durante la etapa de pre-entrenamiento de BERT en la tarea para predecir si el texto A es precedido por el texto B o no, dicho preentrenamiento es de beneficio para el ajuste fino en tareas de NLP (Devlin et al., 2019). A partir de esta representación, BERT genera tres vectores para analizar la secuencia de entrada, conocidos como Embeddings de entrada, como se muestra en la Figura 6 (Devlin et al., 2019):

- Identificadores: Son lo identificadores asignados a la secuencia de entrada de acuerdo con el vocabulario de 30000 identificadores admitidos por BERT.
- Posicionales: Representan la posición en la que se encuentra cada palabra.
- Segmento: Indican si la palabra pertenece a secuencia A o secuencia B.

| Entrada | [CLS] | my | dog | is | play | ##ing | [SEP] | my | cat | is | cute | [SEP] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Identificad | [CLS] | my | dog | is | play | ##ing | [SEP] | my | cat | is | cute | [SEP] |
| Segmento | A | A | A | A | A | A | A | B | B | B | B | B |
| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

Figura 6. Ejemplo de entrada y la representación generada por BERT
Elaboración propia

El procesamiento de la entrada se lleva a cabo por la capa Embedding y posteriormente los embeddings de entrada alimentan a la capa Encoder, el cual consiste en 12 bloques o capas transformer, como se muestra en la Figura 7. Cada una de estas capas tiene 12 capas de atención y tienen una profundidad de 768,

generando en su salida un vector de Word Embeddings de 768 valores por cada elemento en la secuencia de entrada, incluyendo los identificadores CLS y SEP. Aunque se puede acceder al Word Embeddings de cada capa transformer de manera individual, en el trabajo original de BERT, se utiliza el Word Embeddings de la etiqueta CLS correspondiente a la última capa transformer, para las tareas de clasificación, pues la información capturada en cada capa se acumula con la salida de la capa transformer previa, por lo que a la salida del Encoder se obtiene un Word Embeddings enriquecido por cada una de las capas transformer (Devlin et al., 2019).

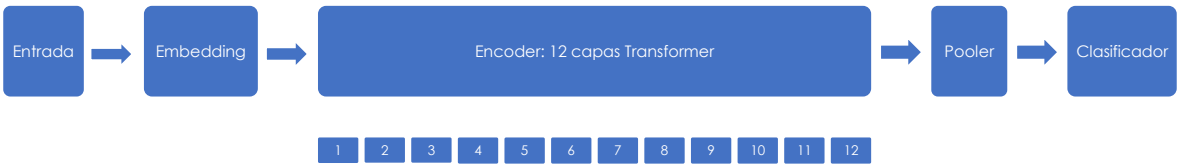



Figura 7. Arquitectura de BERT
Elaboración propia

BERT se entrenó previamente en una base de datos de 3300 millones de palabras (Devlin et al., 2019) para generar estos Word Embeddings en dos tareas: entrenamiento enmascarado y predicción de la siguiente secuencia de texto. En la primera se enmascaran aleatoriamente el 15 % de las palabras en una secuencia de texto con el objetivo de predecir la palabra que debe ocupar ese lugar enmascarado, lo que obliga a BERT a emplear toda la información en la secuencia o contexto para predecir cual es la palabra enmascarada (Devlin et al., 2019). Mientras que en la segunda, se reciben como entrada un par de secuencias de texto con el objetivo de identificar los pares donde la secuencia en el segundo lugar presenta una relación de continuidad con la primera, lo que permite a BERT entender la relación que existe entre dos secuencias (Devlin et al., 2019).

La representación numérica a partir de la salida del encoder (Word Embeddings) se pasa a una capa de linealización conectada a la capa de clasificación, por ejemplo, los autores de BERT usaron una capa de 768 nodos con activación

tangente denominada Pooler para linealizar la salida de BERT hacia una capa de clasificación como se muestra en la Figura 7 (Devlin et al., 2019).

Esta representación numérica (Word Embeddings) es adaptada en la fase de ajuste fino de BERT junto con la capa adicional que será entrenada también, por ejemplo, un clasificador binario basado en una red neuronal prealimentada, mediante el cual se pueda calcular la probabilidad de que la entrada pertenezca a una de dos clases.

En esta tesis se propone un método de transferencia de aprendizaje para el ajuste fino de BERT en una tarea de clasificación binaria de dos secuencias de texto, por lo que a continuación se detalla el ajuste fino estándar de BERT para dicha tarea.

### 2.3.2. Ajuste fino estándar de modelo BERT para clasificación binaria

El proceso de ajuste fino consiste en entrenar el conjunto de parámetros del modelo BERT para proveer la información suficiente a la capa de clasificación para aprender a realizar la nueva tarea objetivo, como se muestra en la **Error! Reference source not found.** en el caso de una clasificación binaria. Particularmente, la importancia de las capas transformer de BERT en el ajuste fino se ha demostrado en estudios previos (Lee et al., 2019), es decir, el desempeño de BERT puede variar dependiendo de las capas transformer que se están entrenando durante el ajuste fino. Esto se debe a que según la profundidad de las capas transformer, la funcionalidad que proveen y la información textual que capturan es diferente (Devlin et al., 2019).

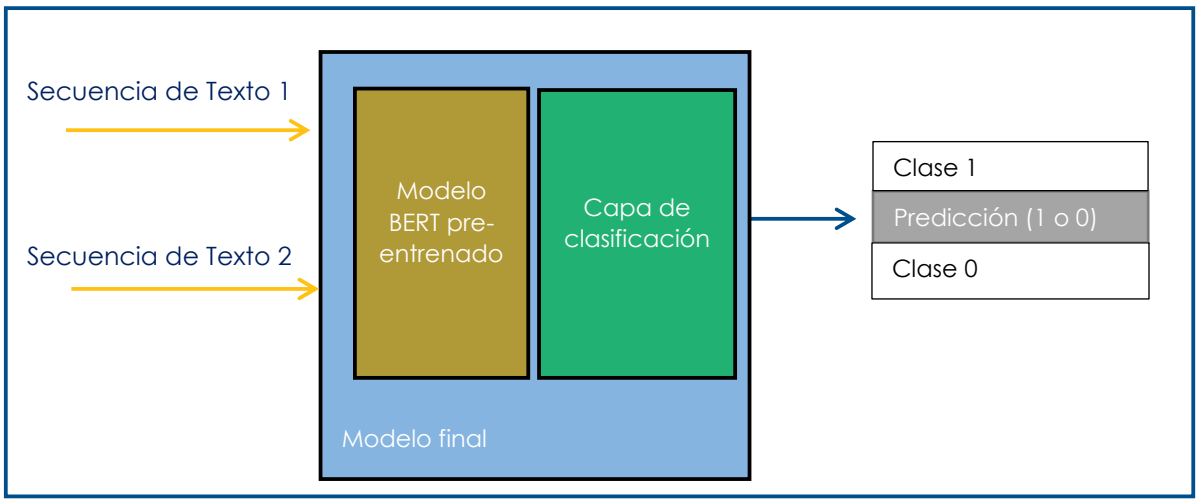



Figura 8. BERT y red neuronal prealimentada como clasificador binario
Elaboración propia

A continuación se analiza la información textual concentrada entre las capas transformer de BERT durante el ajuste fino en tareas de clasificación.

### 2.3.3. Información capturada por las capas transformer en modelo BERT

Para las tareas de clasificación de texto se han hecho experimentos con los Word Embeddings generados por cada capa transformer de manera individual durante un ajuste fino estándar y se ha demostrado que los Word Embeddings de las últimas capas son las que proporcionan un mejor desempeño (Sun et al., 2020).

Particularmente, se ha demostrado que la estructura de la información textual es concentrada en diferentes capas transformer de BERT, es decir, la información semántica se concentra en las últimas capas mientras que la representación sintáctica y la relación entre frases se concentra en las capas intermedias y en las primeras respectivamente (Jawahar et al., 2019). La distribución de esta información textual se puede observar en la Figura 9.


Figura 9. Concentración de información textual en BERT
Elaboración propia

Es posible obtener un desempeño similar o superior, al mismo tiempo que se reduce el ciclo de entrenamiento y los recursos requeridos para hacer la transferencia de aprendizaje, sí la configuración de entrenamiento de las capas transformer es seleccionada adecuadamente (Chen et al., 2023). Para atender esta área de investigación, se han propuesto técnicas de aprendizaje donde se busca identificar las capas transformer y los parámetros que deben ser entrenados en las mismas buscando un equilibrio entre desempeño y en la eficiencia del entrenamiento (Chen et al., 2023). Estas técnicas se describen en la siguiente sección.

## 2.4. TÉCNICAS DE TRANSFERENCIA DE APRENDIZAJE

### 2.4.1. Ajuste fino estándar

El ajuste fino estándar consiste en emplear modelos previamente entrenados complementados por una capa adicional desarrollada para la tarea en específico que se busca llevar a cabo, con una base de datos más pequeña perteneciente al dominio de la tarea en particular (Devlin et al., 2019), por ejemplo, clasificación de texto. Según Devlin et al. (2019) las principales ventajas son:

- No es requerido generar desde cero un modelo propio de representación numérica del lenguaje.
- La base de datos necesaria para el entrenamiento fino es menor.

En la Figura 10 se muestra el concepto de ajuste fino de modelos pre-entrenados para una tarea específica. El modelo de lenguaje recibe la entrada y la capa para la tarea particular está conectada a la salida del modelo de lenguaje.

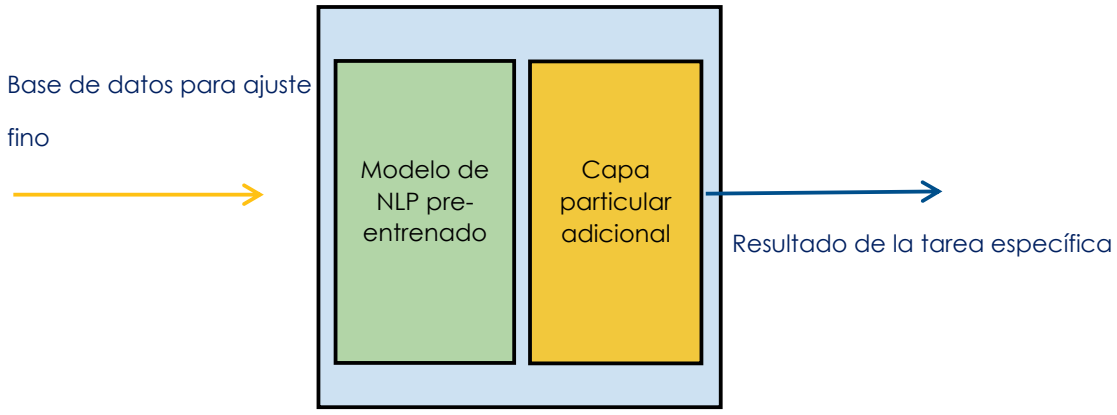



Figura 10. Modelo ajustado finamente
Elaboración propia

### 2.4.2. Entrenamiento eficiente y sondeo lineal

El entrenamiento eficiente de parámetros y capas consiste en ajustar un porcentaje menor de la cantidad total de los parámetros y capas de los que constituyen el modelo, con el objetivo de reducir el tiempo y recursos requeridos para hacer la transferencia de aprendizaje, al mismo tiempo que pretende conservar el desempeño del modelo comparado contra el ajuste fino estándar (Chen et al., 2023; Houlsby et al., 2019; Yang et al., 2022). Estos se pueden dividir en dos tipos, en el primero se fijan capas o parámetros específicos reduciendo la cantidad de parámetros entrenables en el modelo original. Mientras que el segundo, añade capas o parámetros nuevos al modelo original, los cuales son los únicos entrenados conservando fijos todos los parámetros originales del modelo.

Las técnicas representativas de entrenamiento eficiente de parámetros son las siguientes:

- **AutoFreeze (fijación automática):** Automáticamente fija determinadas capas transformer durante el entrenamiento, evaluando el desempeño de cada capa durante los pasos del entrenamiento (Liu et al., 2021).

- **Layer Freezing (capas fijas):** Se fijan de manera arbitraria las capas transformer que no se entrenan, esta selección es motivada por estudios empíricos o por la funcionalidad que las capas han demostrado en estudios previos para la tarea objetivo (Lee et al., 2019).
- ***Bitfit*:** Se entrenan parcialmente los parámetros de cada capa transformer de manera fija, específicamente se entrenan los correspondientes a las ponderaciones de sesgo (bias) (Ben Zaken et al., 2022).
- **Adapters (adaptadores):** Consiste en añadir parámetros mínimos para ser entrenados, mientras que las capas transformer y el resto de parámetros originales permanecen fijados (Houlsby et al., 2019).

Por otro lado, sondeo lineal (Linear probing), busca reducir el tiempo y recursos requeridos para hacer la transferencia de aprendizaje entrenando solamente la capa de clasificación (Yang et al., 2022). Esto lo hace altamente dependiente a la capacidad del modelo de capturar la información textual como resultado del pre-entrenamiento (Kumar et al., 2022).

En esta tesis se propone emplear el método de capas fijas (Layer Freezing) con el objetivo de agrupar la selección propuesta de capas de BERT arbitrariamente de acuerdo a la concentración de información semántica y sintáctica identificada en Jawahar et al. (2019), la cual se explica de manera detallada en la sección

# 3. MÉTODO PROPUESTO

# 3. MÉTODO PROPUESTO

En este capítulo se discute el método propuesto de transferencia de aprendizaje en dos etapas del modelo BERT añadiendo una capa de clasificación binaria para la rastreabilidad entre requerimientos de software descrito de manera general en la Figura 11. La primera etapa se denomina "Enfoque en la información semántica" y la segunda se denomina "Enfoque en información sintáctica y la identificación de frases".

En ambas etapas se emplea la base de datos en la sección 3.4. BASE DE DATOS ESTUDIADA CON EL MÉTODO PROPUESTO y se entrenan la capa Pooler y la capa del clasificador, por lo que solo hay variación en el entrenamiento de las capas transformer entre cada una de las etapas, tal como se describe en el algoritmo de transferencia de aprendizaje en la sección 3.3. ALGORITMO DE TRANSFERENCIA DE APRENDIZAJE DEL MÉTODO PROPUESTO

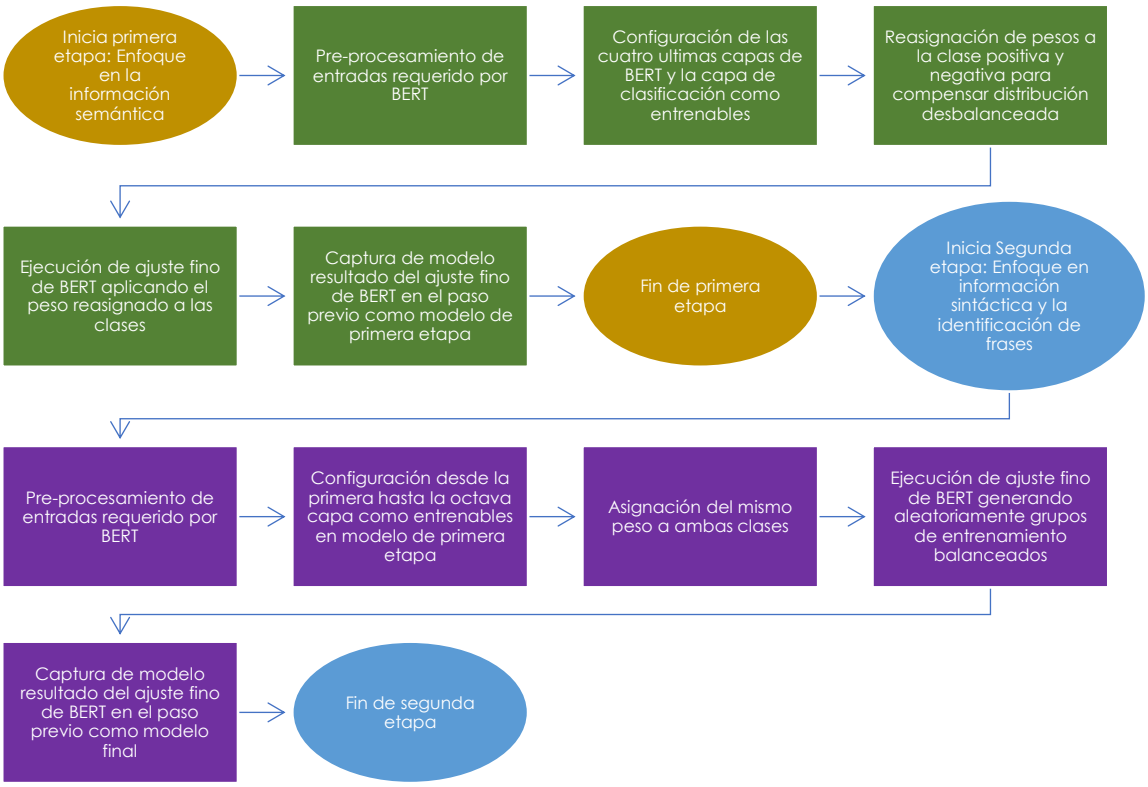



Figura 11. Diagrama de flujo del método propuesto
Elaboración propia

Además, se describen las técnicas para mitigar el desbalance a emplear en el método propuesto: la reponderación de peso de manera temprana en la primera etapa y el remuestreo en la segunda etapa para la automatización de la rastreabilidad de requerimientos de software de distintos niveles. Estudios previos han demostrado teórica y empíricamente la importancia de mitigar el efecto de un entrenamiento desbalanceado en el ámbito general de los modelos de lenguaje profundo (ValizadehAslani et al., 2022) y en la rastreabilidad de requerimientos de software(Lin et al., 2021, 2022; Tian et al., 2023).

## 3.1. PRIMERA ETAPA: ENFOQUE EN LA INFORMACIÓN SEMÁNTICA

En la primera etapa el entrenamiento se configura solamente en el grupo de capas transformer que concentran la representación semántica (Capas entrenables) y configura como no entrenables el resto de las capas (Capas fijas), como se muestra en la Figura 12. Además, se aplica la reponderación de peso de las clases en la función de costo de acuerdo con el inverso de la frecuencia de cada clase escalado al 50 %.

El objetivo de esta primera etapa es que el modelo se concentre en capturar el significado de los requerimientos al mismo tiempo que aprende a darle mayor importancia a los casos de traza verdadera por medio de reponderación de pesos. Además de proporcionar una mitigación al sobre ajuste (Shi et al., 2022).

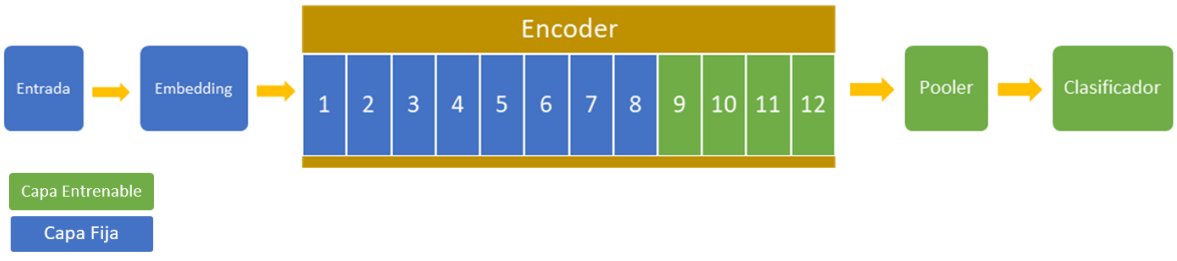



Figura 12. Primera etapa de método propuesto
Elaboración propia

## 3.2. SEGUNDA ETAPA: ENFOQUE EN INFORMACIÓN SINTÁCTICA Y LA IDENTIFICACIÓN DE FRASES

En la segunda etapa se configuran como entrenables solamente el grupo de capas transformer que concentran la información sintáctica y la relación entre frases (Capas entrenables) y se configura como no entrenables el resto de las capas (Capas fijas). Es decir, solo se entrenan desde la capa 1 hasta la capa 8, acorde la Figura 13. Al mismo tiempo, se compensa el tamaño reducido de la base de datos generando dinámicamente paquetes balanceados de entrenamiento.

El objetivo de esta etapa es que el modelo se concentre en identificar la relación entre frases, actores, objetos y acciones en los requerimientos al mismo tiempo que se mitiga la poca disponibilidad de muestras de trazas verdaderas con remuestreo.

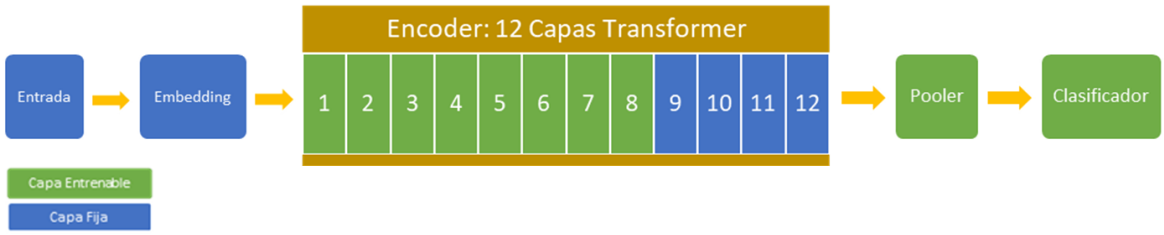



Figura 13. Segunda etapa de método propuesto
Elaboración propia

## 3.3. ALGORITMO DE TRANSFERENCIA DE APRENDIZAJE DEL MÉTODO PROPUESTO

El método se describe a continuación de acuerdo con la Figura 11:

1. Inicia primera etapa: Enfoque en la información semántica.
2. Se preprocesa el grupo de entrenamiento y evaluación de acuerdo con el formato requerido por BERT.
3. Se establece la reponderación del peso de cada clase en la función de costo:

   a. Peso clase 0 es igual al inverso de la frecuencia de la clase 0 escalado al 50 %
   b. Peso clase 1 es igual al inverso de la frecuencia de la clase 0 escalado al 50 %
4. Se especifican las cuatro últimas capas de BERT como las únicas entrenables y se fija el resto.
5. Se inicia el ciclo de la primera etapa de ajuste fino hasta completar el número de ciclos n.
6. Se captura el modelo como modelo de primera etapa.
7. Inicia segunda etapa: Enfoque en información sintáctica y la identificación de frases.
8. Se preprocesa el grupo de entrenamiento y evaluación de acuerdo con el formato requerido por BERT.
9. En la función de costo se asigna el mismo peso a todas las clases.
10. Se especifican desde la primera hasta la octava capa de BERT como las únicas capas entrenables y se fija el resto a partir del modelo de primera etapa.
11. Se inicia el ajuste fino de la segunda etapa aplicando un remuestreo aleatorio para generar paquetes balanceados de muestras durante un ciclo:
    a. 50 % son casos trazas verdades.
    b. 50 % es de trazas falsas.
12. Se captura el modelo como modelo final.

## 3.4. BASE DE DATOS ESTUDIADA CON EL MÉTODO PROPUESTO

La base de datos a emplear para entrenar y probar el modelo en esta tesis está compuesta por un grupo de requerimientos de software del proyecto CM1 de la NASA (Sundaram et al., 2005) disponible de manera pública en la página de CoEST (center of excellence for software traceability) (*CoEST*, s/f). La base de datos está dividida por dos tipos de requerimientos: bajo nivel y alto nivel. Entre determinados pares de elementos de ambos grupos de requerimientos existe una relación de rastreabilidad a la cual por simplicidad de la redacción de este documento denominaremos traza. Por lo tanto, al analizar la relación entre un par de ambos

tipos de requerimientos encontramos elementos que sí tienen traza y otros que no tienen traza. A partir del razonamiento previo, definiremos en este documento a una muestra como un par de requerimientos, uno de alto nivel y uno de bajo nivel, asimismo, según la relación de rastreabilidad que existe entre el par de requerimientos en una muestra, ésta pertenecerá solamente a una de las dos siguientes clases:

- **Traza Verdadera**: Relación válida de rastreabilidad.
- **Traza Falsa**: No existe relación de rastreabilidad.

La base de datos tiene 53 requerimientos únicos de bajo nivel y 22 únicos de alto nivel, generando un total de 1166 muestras posibles, de las cuales previamente se etiquetaron un total de 45 como trazas verdaderas, lo que da como resultado un total de 1121 trazas falsas. Como se puede observar, ésta es una base de datos no balanceada, pues la cantidad de muestras que pertenecen a la clase de trazas falsas es mayor que la clase de trazas verdaderas, dicho comportamiento, es común en los proyectos de software (J. Guo et al., 2017).

Tabla 3. Ejemplos de traza verdadera (clase 1) y traza falsa (clase 0) de la base de datos CM1

| Requerimientos de alto nivel | Requerimientos de bajo nivel | Clase |
|---|---|---|
| The DPU-CCM shall be able to count a consecutively reported error. When the count for a particular error ID, exceeds 250 for a particular reporting period, the error code will be replaced with an error code sequence which shall include the original error code and the number of times the error was reported. | Error Collection and Reporting The S_ccm_ERR_REPEAT error encodes the count of the last repeated error in its low order byte. If a new error is reported as discussed above, ccmErrEnq() will enqueue a S_ccm_ERR_REPEAT for any previously repeated error, along with the newly reported error. In order to keep the original error codes and their repeated counts together in the same error packet, ccmMkHkErr(), enqueues a special error code, S_ccm_ERRQ_FLUSH, as a special signal to ccmErrEnq() that it needs to clear its error tracking mechanism and enqueue any repeated error counts associated with a particular error. | 1 |
| The DPU-CCM shall be able to count a consecutively reported error. When the count for a particular error ID, exceeds 250 for a particular reporting period, the error code will be replaced with a error code sequence which shall include the original error code and the number of times the error was reported. | Ring buffer data structure A separate pointer indicates the end of the buffer: pEnd. This pointer points to the last available location in the buffer. Before advancing either pointer it should be verified whether the pointer points to the last location, in that case the pointer is reset to the start of the buffer, pointed to by pBuf, otherwise it can just be incremented. | 0 |

(Sundaram et al., 2005)

## 3.5. MARCO DE COMPARACIÓN

Se elaboró un marco de comparación entre el método propuesto en esta tesis y las siguientes metodologías de transferencia de aprendizaje en la tarea de clasificación binaria para identificar las trazas verdaderas en un grupo de candidatos de pares de requerimientos evaluando los resultados en términos de F2:

- Ajuste fino estándar: Entrenamiento de todas las capas transformer de BERT, es decir el 100 % del modelo.
- Ajuste fino fijado al 50 %: Las primeras seis capas transformer de BERT se fijan en el ajuste fino, es decir, solamente se configuran como entrenables el 50 % de capa transformer de BERT.

- Sondeo lineal (Linear probing): Ninguna capa transformer de BERT es entrenada.

Cabe aclarar que las capas de clasificación binaria y Pooler se entrenaron en todos los experimentos, excepto en el método de sondeo lineal donde sólo se entrenó la capa de clasificación.

### 3.5.1. Modelo para clasificación de rastreabilidad de requerimientos

El modelo está constituido por dos bloques, el primero es el modelo de BERT pre-entrenado y el segundo un clasificador que determina el resultado de la clasificación como positiva en el caso de que se prediga una relación entre ambos requerimientos o negativa en el caso contrario, como se muestra en la Figura 14. El mismo modelo se entrenó en todos los métodos en el marco de comparación.

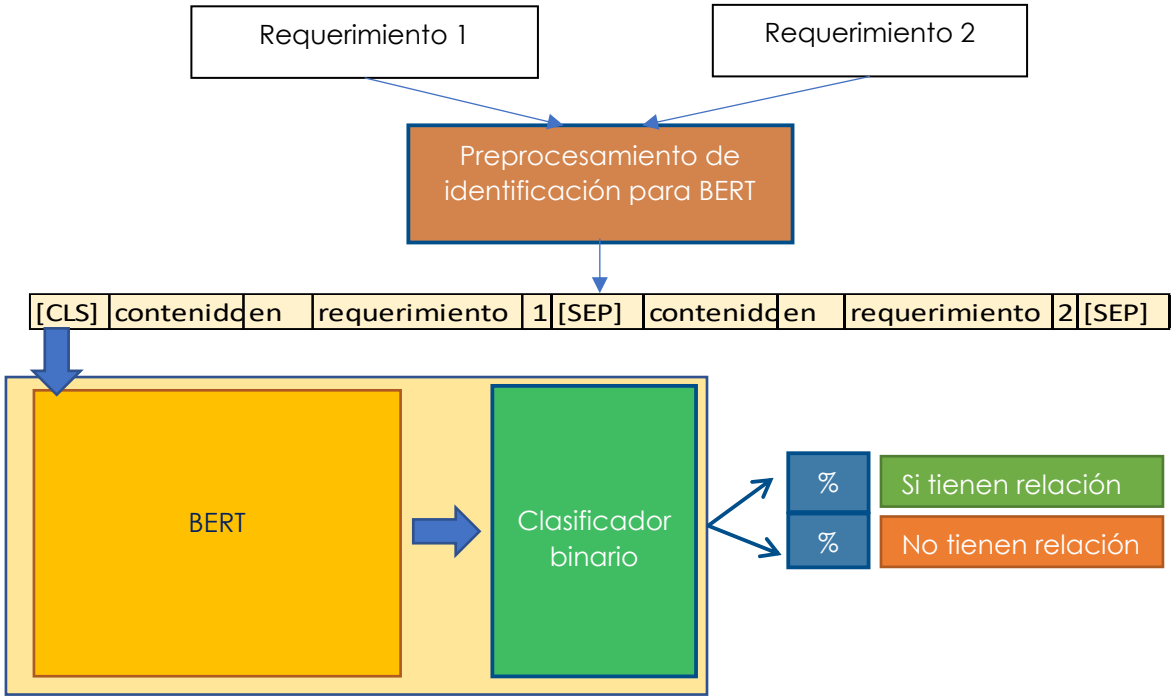



Figura 14. Arquitectura del modelo de clasificación
Elaboración propia

#### 3.5.1.1. Arquitectura del modelo de clasificación

A continuación se describe de manera general la arquitectura del funcionamiento del modelo como se ilustra en la Figura 14. Este modelo se entrenó tanto con el método propuesto como con los métodos de referencia:

1. Se ingresan dos requerimientos en lenguaje natural.
2. Se preprocesan ambos requerimientos para que sean recibidos en el formato esperado por BERT.
3. BERT analiza ambos requerimientos y genera la representación numérica de su relación semántica en un vector de 768 elementos.
4. El clasificador basado en una red neuronal prealimentada recibe la representación numérica y la procesa para predecir si tienen o no relación ambos requerimientos.
5. Se normaliza la predicción de la red neuronal para determinar la predicción final.

El clasificador está compuesto de una capa prealimentada con dos nodos de salida, uno por cada clase. En el modelo base de BERT empleado en este estudio existe una capa prealimentada de 768 nodos con activación tangente que linealiza la salida de BERT para pasarla a la capa del clasificador que posteriormente será normalizado para obtener un porcentaje de probabilidad de cada clase, como se muestra en la Figura 15.

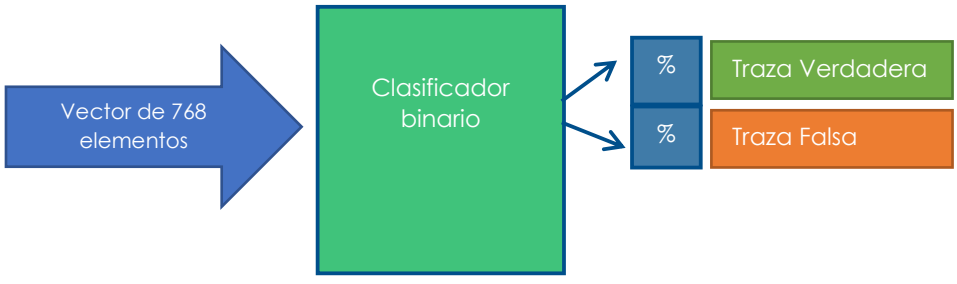



Figura 15. Linealización hacia capa de clasificación
Elaboración propia

#### 3.5.1.2. Plataforma de implementación

El modelo fue implementado en Tensorflow (Abadi et al., 2016), pues provee una serie de bloques predefinidos que permiten agilizar el desarrollo de modelos de NLP,

así como modelos pre-entrenados. A continuación se enlistan los módulos de Tensorflow (Abadi et al., 2016) que fueron empleados:

- Modelo BERT pre-entrenado: Es la implementación de BERT de los autores con el objetivo de ser empleado en aplicaciones de transferencia de aprendizaje.
- Adam optimizer: Es el optimizador de Adam para entrenar el modelo BERT de acuerdo con el trabajo original.
- Cross entropy loss function: La función entropía cruzada (Cross entropy loss function en inglés) es empleada como función de costo para calcular el error entre la predicción del modelo y el valor real.

### 3.5.2. Criterio de evaluación

Se empleó la técnica de doble validación cruzada para el entrenamiento y la evaluación de todos los modelos: una validación cruzada externa (VCE) y una validación cruzada interna (VCI). La VCE consistió en una validación cruzada de cuatro subconjuntos, es decir, se particionó la base de datos en cuatro grupos no superpuestos conservando la distribución original de las clases (3.85 % trazas verdaderas), de los cuales tres conjuntos se destinaron como grupos de entrenamiento y uno como grupo de prueba. Subsecuentemente, para la VCI se particionó el grupo de entrenamiento en tres subconjuntos no superpuestos, de igual manera preservando el balance original entre clases, dejando dos para el entrenamiento interno y uno de evaluación. La VCI se utilizó para la optimización de hiperparámetros de cada modelo con base en los resultados en el grupo de evaluación y la VCE para obtener el promedio de las métricas de F2, recuperación y precisión en la ejecución del grupo de prueba por cada modelo.

En la **Error! Reference source not found.** se muestra de manera general la doble validación cruzada con el desglose de las particiones de la base de datos para cada una de las cuatro rondas externas y sus tres rondas internas respectivas.

| | | | | |
|---|---|---|---|---|
| VCE ronda 1 | Entrenamiento externo | Entrenamiento externo | Entrenamiento externo | Prueba |
| VCI ronda 1 | Entrenamiento interno | Entrenamiento interno | Evaluación | |
| VCI ronda 2 | Entrenamiento interno | Evaluación | Entrenamiento interno | |
| VCI ronda 3 | Evaluación | Entrenamiento interno | Entrenamiento interno | |
| | | | | |
| VCE ronda 2 | Entrenamiento externo | Entrenamiento externo | Prueba | Entrenamiento externo |
| VCI ronda 1 | Entrenamiento interno | Entrenamiento interno | | Evaluación |
| VCI ronda 2 | Entrenamiento interno | Evaluación | | Entrenamiento interno |
| VCI ronda 3 | Evaluación | Entrenamiento interno | | Entrenamiento interno |
| | | | | |
| VCE ronda 3 | Entrenamiento externo | Prueba | Entrenamiento externo | Entrenamiento externo |
| VCI ronda 1 | Entrenamiento interno | | Entrenamiento interno | Evaluación |
| VCI ronda 2 | Entrenamiento interno | | Evaluación | Entrenamiento interno |
| VCI ronda 3 | Evaluación | | Entrenamiento interno | Entrenamiento interno |
| | | | | |
| VCE ronda 4 | Prueba | Entrenamiento externo | Entrenamiento externo | Entrenamiento externo |
| VCI ronda 1 | | Entrenamiento interno | Entrenamiento interno | Evaluación |
| VCI ronda 2 | | Entrenamiento interno | Evaluación | Entrenamiento interno |
| VCI ronda 3 | | Evaluación | Entrenamiento interno | Entrenamiento interno |

Figura 16. Doble validación cruzada empleada en la experimentación
Elaboración propia

La doble validación cruzada con su correspondiente generación de particiones se repitió cincos veces con semillas diferentes y se promedió el resultado de las cinco ejecuciones como el puntaje final de F2 para comparar todos los modelos.

### 3.5.3. Configuración de entrenamiento

A continuación se describen las técnicas empleadas para mitigar el desbalance de clases y los detalles de entrenamiento de cada modelo.

#### 3.5.3.1. Reponderación de peso de clases

La técnica de reponderación de peso de las clases se usó para compensar el desbalance entre las clases en todos los modelos generados con los métodos de referencia y en la primera etapa del método propuesto, dándole un mayor peso en la función de costo a cada clase de acuerdo con el inverso de la frecuencia de cada clase en la base de datos escalado al 50 %:

- Peso traza verdadera (Clase 1): 12.96.
- Peso traza falsa (Clase 0): 0.52.

#### 3.5.3.2. Balance aleatorio de paquetes de entrenamiento

En la segunda etapa del método propuesto se generaron aleatoriamente paquetes balanceados de 32 muestras en cada iteración de entrenamiento (H.

Guo et al., 2018), de tal manera que cada paquete contenía 16 muestras de trazas falsas y 16 muestras de trazas verdaderas.

#### 3.5.3.3. Modelo de referencia con ajuste fino estándar

Este modelo se denominó como AFE, se configuraron todas las capas como entrenables, como se muestra en la Figura 17, y se procedió a llevar a cabo los pasos de experimentación descritos en la sección 3.5.3.7. **Pasos para la ejecución de los experimentos** de acuerdo con los hiperparámetros descritos en la Tabla 4.

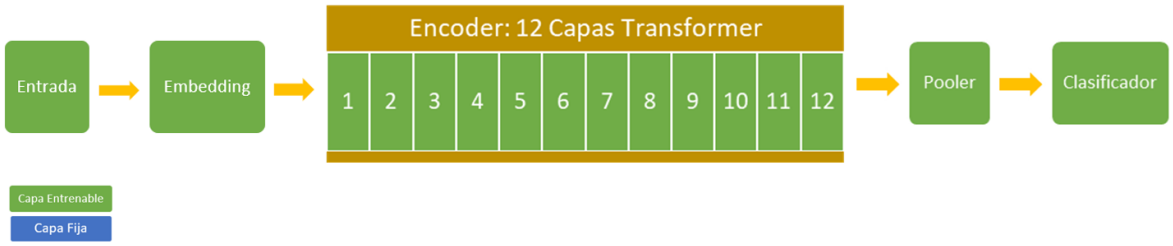



Figura 17. Capas entrenables en modelo de referencia con ajuste fino estándar
Elaboración propia

Tabla 4. Parámetros de búsqueda de optimización de ajuste fino estándar

| Parámetro | Valor |
|---|---|
| Tasa de aprendizaje | 1e-5, 2e-5, 5e-5 |
| Numero de ciclos (epochs) | 1,3,5 |
| Tamaño de los paquetes | 32 |
| Optimizador | Adam |

Elaboración propia

#### 3.5.3.4. Modelo de referencia con ajuste fino fijo al 50 %

Este modelo se denominó AFF, se configuraron como entrenables desde la séptima capa Transformer en adelante fijando el resto de las capas, es decir, el 50 % de las capas, como se muestra en la Figura 18. Después, se procedió a llevar a cabo los pasos de experimentación descritos en la sección 3.5.3.7. **Pasos para la ejecución de los experimentos**de acuerdo con los hiperparámetros descritos en la Tabla 5.


Figura 18. Capas entrenables en modelo de referencia con ajuste fino fijo (50 %)
Elaboración propia

Tabla 5. Parámetros de búsqueda de optimización de ajuste fino fijo (50 %)

| Parámetro | Valor |
|---|---|
| Tasa de aprendizaje | 1e-2, 1e-3, 1e-4 |
| Numero de ciclos (epochs) | 1,3,5 |
| Tamaño de los paquetes | 32 |
| Optimizador | Adam |

Elaboración propia

### 3.5.3.5. Modelo de referencia de sondeo lineal

Este modelo se denominó LP, se fijaron todas las capas del modelo entrenando solamente la capa de clasificación, como se muestra en la Figura 19. Después, se procedió a llevar a cabo los pasos de experimentación descritos en la sección 3.5.3.7. Pasos para la ejecución de los experimentosde acuerdo con los hiperparámetros descritos en la Tabla 6.

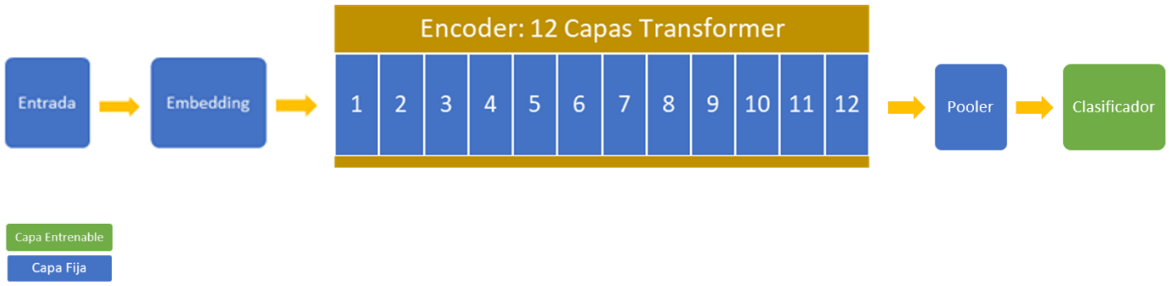



Figura 19. Capas entrenables en modelo de referencia con sondeo lineal
Elaboración propia

Tabla 6. Parámetros de búsqueda de optimización de modelo ajuste fino fijo

| Parámetro | Valor |
|---|---|
| Tasa de aprendizaje | 1e-2, 1e-3, 1e-4 |
| Numero de ciclos (epochs) | 1,3,5 |
| Tamaño de los paquetes | 32 |

| | |
|---|---|
| Optimizador | Adam |

Elaboración propia

#### 3.5.3.6. Modelo con método propuesto

Este modelo se denominó MP, se siguió el algoritmo descrito en la sección 3.3. ALGORITMO DE TRANSFERENCIA DE APRENDIZAJE DEL MÉTODO PROPUESTO y en cada etapa se procedió a llevar a cabo los pasos de experimentación descritos en la sección 3.5.3.7. Pasos para la ejecución de los experimentos de acuerdo con los hiperparámetros descritos para la primera etapa en la Tabla 7 y para la segunda etapa en la Tabla 8.

Tabla 7. Parámetros de búsqueda de optimización en la primera etapa del método propuesto

| Parámetro | Valor |
|---|---|
| Tasa de aprendizaje | 1e-2, 1e-3, 1e-4 |
| Numero de ciclos (epochs) | 1,3,5 |
| Tamaño de los paquetes | 32 |
| Optimizador | Adam |

Elaboración propia

Tabla 8. Parámetros de búsqueda de optimización en la segunda etapa del método propuesto

| Parámetro | Valor |
|---|---|
| Tasa de aprendizaje | 1e-5, 2e-5 |
| Numero de ciclos (epochs) | 1 |
| Tamaño de los paquetes | 32 |
| Optimizador | Adam |

Elaboración propia

#### 3.5.3.7. Pasos para la ejecución de los experimentos

Se entrenaron todos los modelos siguiendo estos pasos:

1. Se preprocesaron los grupos de entrenamiento externo, entrenamiento interno, evaluación y prueba de la base de datos para cada ronda de VCI y VCE de acuerdo con la distribución original de las clases y de acuerdo con el formato requerido por BERT.
2. Se inicializó la optimización de hiperparámetros mediante la búsqueda exhaustiva de acuerdo con los hiperparámetros para el modelo en la VCI.
3. Se inició el ciclo de entrenamiento del modelo hasta completar la VCI.

4. Se seleccionaron los hiperparámetros con mayor desempeño en F2 en la VCI para emplear en las respectivas rondas de VCE.
5. Se inició el ciclo de entrenamiento del modelo hasta completar la VCE.
6. Se obtuvo el promedio de F2, recuperación y precisión como resultado de las rondas del VCE.
7. El proceso se repitió un total de cinco veces con una semilla diferente.

# 4. RESULTADOS

En esta sección se describen los resultados obtenidos y se analiza el desempeño del modelo generado con el método propuesto (MP) en términos de F2 en el marco de comparación definido anteriormente.

## 4.1 DESCRIPCIÓN DE LOS RESULTADOS

Por motivos ilustrativos en la Figura 20 se muestra una captura de los resultados de la ejecución completa de la VCI correspondiente a la cuarta ronda de la VCE del modelo propuesto en una de las semillas, donde se puede observar que la taza de aprendizaje de 1e-4 en combinación con 3 ciclos de entrenamiento permitieron obtener el puntaje mayor de F2 (0.404) entre todas las combinaciones posibles en la búsqueda de hiperparámetros dentro de la VCI. Consecuentemente, en la Figura 21 y Figura 22 se presentan las gráficas y la captura de la ejecución de la cuarta ronda de VCE empleando dicha configuración respectivamente, donde se observa el puntaje de F2, recuperación (recall), precisión y el error (loss).

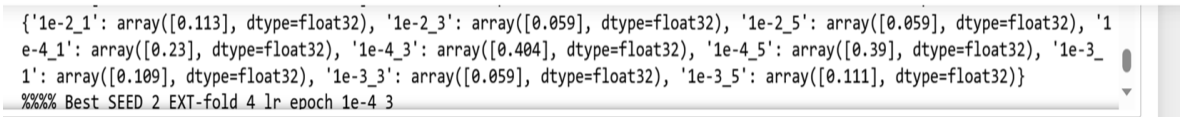



Figura 20. Ejecución completa de VCI correspondiente a una ronda de VCE
Elaboración propia

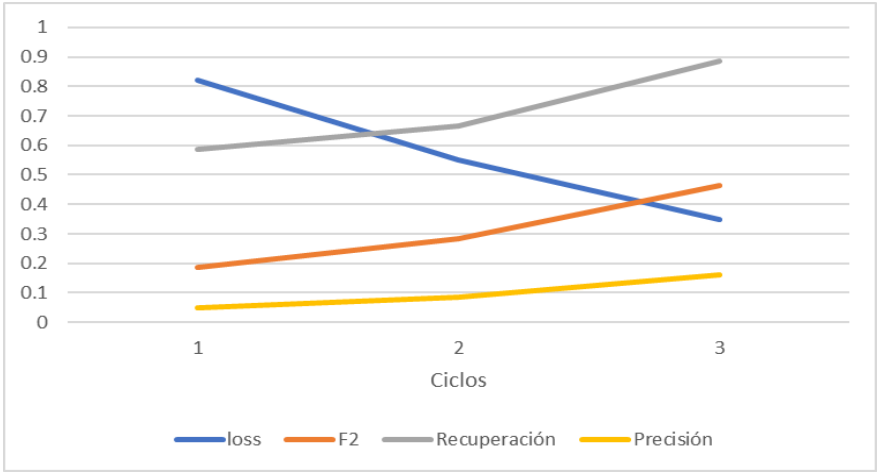



Figura 21. Gráficas de entrenamiento en ronda de VCE
Elaboración propia

```
Epoch 1/3
27/27 [==============================] - 31s 749ms/step - loss: 0.8225 - f2: 0.1883 - recall: 0.5882 - precision: 0.0506
Epoch 2/3
27/27 [==============================] - 20s 757ms/step - loss: 0.5514 - f2: 0.2842 - recall: 0.6667 - precision: 0.0863
Epoch 3/3
27/27 [==============================] - 20s 753ms/step - loss: 0.3536 - f2: 0.4655 - recall: 0.8857 - precision: 0.1606
```

Figura 22. Captura de entrenamiento en ronda de VCE
Elaboración propia

En la Figura 23 se muestra el rango del valor de F2 para cada modelo, obtenido como resultado de la doble validación cruzada con cada una de las cinco semillas. Así mismo, en la Figura 23 se puede apreciar que el modelo con el límite superior mayor de F2 es AFE (0.535), seguido por MP (0.501), pero, AFE es el modelo que presenta un mayor rango en sus resultados, pues su límite inferior (0.236) tiene una diferencia de 6 milésimas contra el límite inferior más bajo (0.23) y de 135 milésimas contra el límite inferior más alto (0.371), presentados por LP y MP, respectivamente.

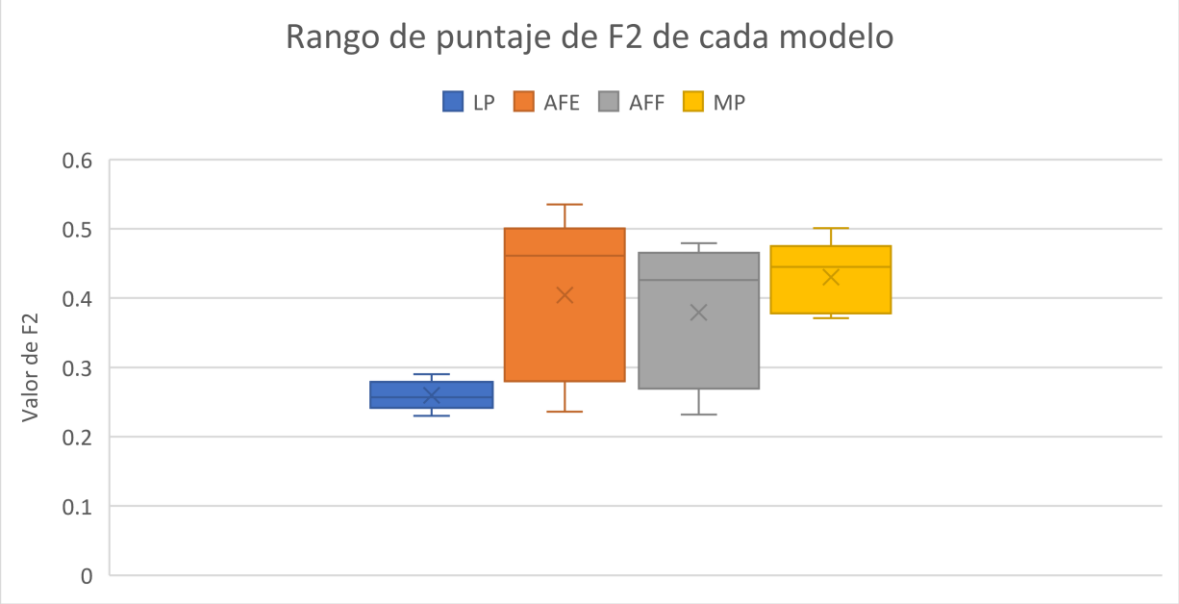



Figura 23. Rango de valores del puntaje de F2 de cada modelo por semilla
Elaboración propia

En la Tabla 9 se muestra el puntaje de F2, recuperación (R) y precisión (P) reportado por cada modelo obtenido del valor promedio con las cinco semillas, además, se calcularon las métricas de verdaderos positivos (TP*), verdaderos negativos (TN*), falsos positivos (FP*) y falsos negativos (FN*) de acuerdo con los resultados promedios proyectados en el grupo de prueba de 292 trazas (11 trazas verdaderas y 281 trazas falsas). El mayor puntaje en F2 y precisión fue obtenido por MP, 0.4302

y 0.2504 respectivamente, mientras que LP fue el que obtuvo el mayor puntaje en recuperación (0.9258), pero contrastantemente, también presenta el menor puntaje en F2 y precisión. Adicionalmente, en la Figura 24 se puede observar la comparación de F2, recuperación y precisión entre todos los modelos.

Tabla 9. Resultados promedio de la doble validación cruzada.

| Modelo | F2 | R | P | TP* | TN* | FP* | FN* | (TP*+FP*) |
|---|---|---|---|---|---|---|---|---|
| MP | 0.4302 | 0.5518 | 0.2504 | 6 | 263 | 18 | 5 | 24 |
| AFE | 0.4044 | 0.6304 | 0.1782 | 7 | 249 | 32 | 4 | 39 |
| AFF | 0.3792 | 0.601 | 0.1696 | 7 | 249 | 32 | 4 | 39 |
| LP | 0.2596 | 0.9258 | 0.0702 | 10 | 146 | 135 | 1 | 145 |

Elaboración propia

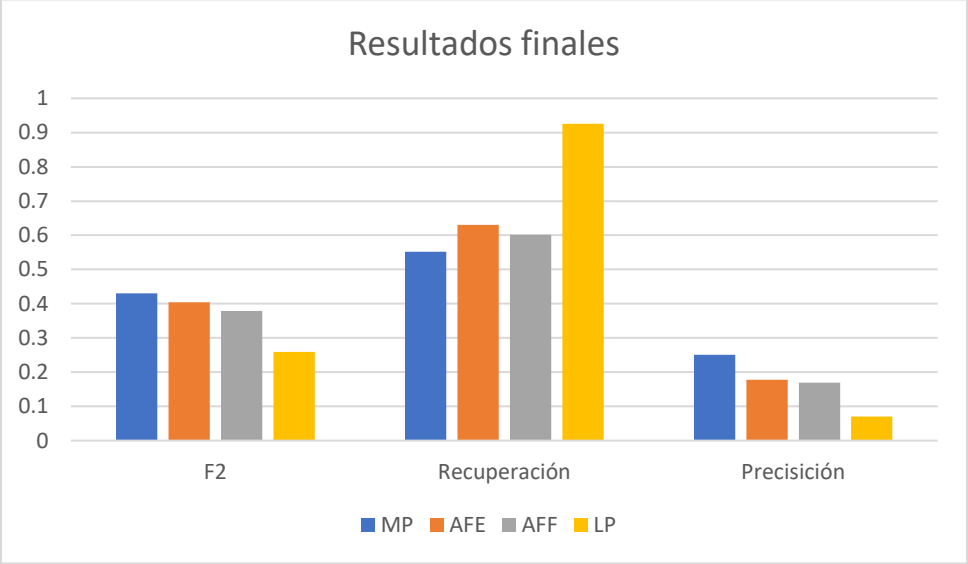
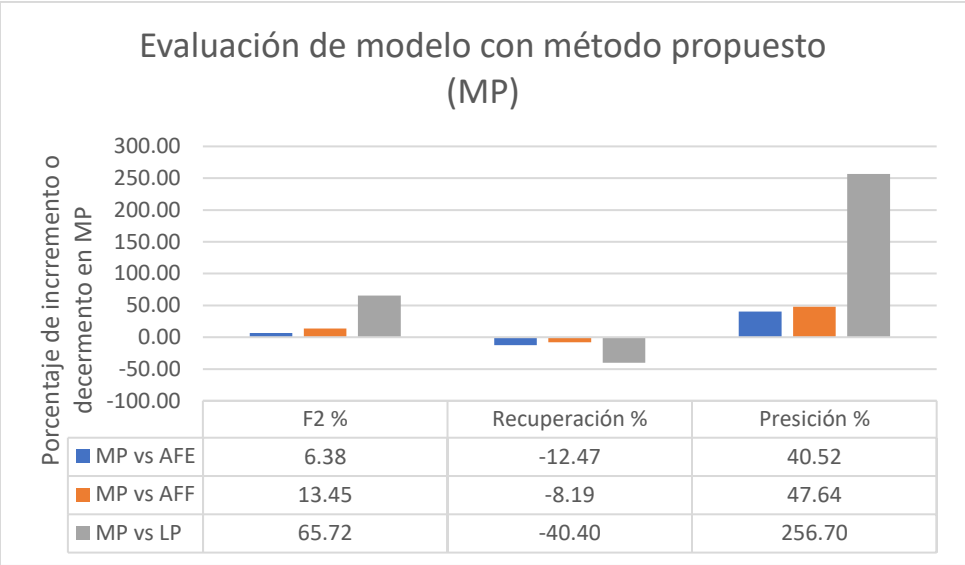



Figura 24. Comparación de F2, recuperación y precisión.
Elaboración propia

Mientras que en la Figura 25 se muestra la diferencia en porcentaje obtenida en las métricas evaluadas contra MP.




| | F2 % | Recuperación % | Presición % |
|---|---|---|---|
| MP vs AFE | 6.38 | -12.47 | 40.52 |
| MP vs AFF | 13.45 | -8.19 | 47.64 |
| MP vs LP | 65.72 | -40.40 | 256.70 |

Figura 25. Porcentaje de incremento o decremento en MP
Elaboración propia

## 4.2 ANÁLISIS DE RESULTADOS

El análisis de los resultados de F2 indica que el modelo entrenado con el método propuesto tiene una mejora del 6.38 % con respecto al modelo AFE (ver Figura 25), lo cual cumple con la hipótesis planteada en esta tesis respaldando que es posible obtener un incremento del 1 % en F2 contra el método estándar de ajuste fino empleando el método propuesto. Es decir, la mejora se logra entrenando las capas que concentran la información semántica en una primera etapa y entrenando las capas transformer que capturan las relaciones sintácticas y a nivel frase en una segunda etapa. Además, es apreciable que hay un incremento en la métrica de F2 del 13.45 % respecto al modelo AFF (ajuste fino fijo del 50 % del modelo), lo cual cumple con la hipótesis planteada en esta tesis respaldando que es posible obtener un incremento del 1 % en F2 contra el método ajuste fino al 50 % empleando el método propuesto. Por otra parte, es congruente con investigaciones previas de que el ajuste fino fijo puede presentar un decremento contra el ajuste fino estándar (Lee et al., 2019; Liu et al., 2021). Adicionalmente, el modelo entrenado con el método propuesto presenta un incremento en la métrica de F2 del 65.72 % sobre el método de LP, por lo tanto, respalda la hipótesis planteada en esta tesis sustentando que es posible obtener un incremento del 1 % en F2 contra el método

LP empleando el método propuesto. De igual manera, esta diferencia es respaldada por el hecho de que LP puede presentar un desempeño menor comparado con el ajuste fino estándar si el modelo pre-entrenado no fue entrenado en un contexto similar a la base de datos objetivo, por ejemplo requerimientos de software, lo cual es el caso de BERT (Yang et al., 2022). Adicional al alcance de esta tesis, también se calcularon los resultados de recuperación, precisión, TP, TN, FP y FN, debido a que exponen una manera complementaria de analizar los resultados.

En términos de recuperación, el modelo entrenado con el método propuesto presenta un decremento en comparación de todos los modelos, como se muestra en la Figura 25, particularmente contra la técnica de LP, presentando un decremento del 40.40 %. Esta diferencia es explicable a la relación que existe entre precisión y recuperación, pues si la precisión de un clasificador binario sube es posible que al mismo tiempo la recuperación baje (Laliberte et al., 2022). Esto sucede en este escenario, ya que el modelo entrenado con el método propuesto tiene un incremento sobre los otros modelos en términos de precisión, especialmente contra LP, presentando un incremento del 256.70 %. Por otra parte, el puntaje de LP en precisión (0.0702) en la Tabla 9 es un indicador de que tiende a clasificar a las muestras como trazas verdaderas, lo que provoca un aumento en la cantidad de FP, que en consecuencia, explica que sea el modelo con puntaje F2 más bajo. Por lo tanto, es posible observar que la precisión puede incrementarse si la capacidad del modelo en la identificación de los actores, objetos y acciones en los requerimientos es enfatizada por medio del entrenamiento de las capas que capturan la información sintáctica y por el remuestreo para balancear las clases cómo describe durante la segunda etapa del método propuesto. Aunque la capacidad de recuperación de las clases positivas es el principal objetivo del modelo entrenado con el método propuesto, se considera relevante tomar en cuenta la precisión para evitar caer en un modelo que clasifique ingenuamente todas las muestras como positivas buscando un valor de recuperación alto. Esto tiene un impacto en el tiempo que el ingeniero de software dedique a descartar los FP de la lista de candidatos proporcionados por el clasificador binario. En la

Tabla 9 se observa que de un grupo de 292 trazas LP proporciona una lista de 145 candidatos, donde solamente 7 son TP, mientras que el modelo entrenado con el método propuesto proporciona una lista de 24 candidatos, donde 6 son TP. En el caso de un ingeniero sin experiencia podría representar una diferencia significativa en la dificultad de la tarea y en el esfuerzo invertido en la misma (Laliberte et al., 2022; Tian et al., 2023).

Finalmente, se observa que el rango de F2 de acuerdo con la Figura 23, se puede representar en centésimas de F2 de 6, 30, 24 y 13 para LP, AFE, AFF y MP, respectivamente. Es destacable que el LP es el modelo con el menor rango de variación en los resultados de F2, esto se puede explicar a la cantidad reducida de capas que son entrenadas en LP (clasificador), pues estudios previos han demostrado que la reducción de capas entrenadas durante el ajuste fino reduce la variación en los resultados (Kumar et al., 2022; ValizadehAslani et al., 2022). Lo cual, también es respaldado por el hecho de que AFE, que es entrenado con todas las capas durante el ajuste fino, es el que presenta el mayor rango de variación de la métrica de F2, seguido por AFF (50 % de capas entrenadas).

# CONCLUSIONES

En esta tesis se llevó a cabo un estudio para proponer un método de transferencia de aprendizaje en dos etapas para la tarea de rastreabilidad de requerimientos de software. El método propuesto considera la concentración de la información semántica y sintáctica entre las distintas capas transformer en BERT, es decir, pretende capturar la relación entre los actores, acciones, objetos, frases, así como el significado de los requerimientos dentro de su contexto en particular (Sonbol et al., 2022b). En la primera etapa se ajustan finamente solo las capas transformer que concentran la información semántica, mientras que en la segunda etapa se entrenan únicamente las capas transformer que concentran la relación entre frases y la información sintáctica.

La evaluación del método propuesto se hizo dentro de un marco de comparación contra tres metodologías de transferencia de aprendizaje usando una base de datos de requerimientos de software de acceso público. Se logró como resultado un incremento mayor al 1 % en el promedio de la métrica de F2 con respecto a los tres métodos previos sustentado con evidencia la hipótesis planteada en esta tesis, obteniendo un incremento de 6.38 %,13.45 % y 65.72 % sobre AFE, AFF y LP respectivamente. Este incremento en F2, es respaldado por la doble validación cruzada, ejecutada cinco veces con semillas diferentes, lo que a su vez permitió observar que el modelo entrenado con el método propuesto presenta el segundo menor rango de variación de resultados (13 centésimas de F2), solo superado por LP (6 centésimas de F2). Esto tiene sentido porque en LP solamente se entrena la capa del clasificador, pero la métrica de F2 obtenida es 65.72 % menor comparada contra el modelo entrenado con el método propuesto.

Adicionalmente, el método propuesto presenta la ventaja de ser una herramienta más confiable en sus predicciones de trazas verdaderas contra el resto de las metodologías dentro del marco de comparación, pues presenta un incremento en precisión 40.52 % sobre el AFE, 47.64 % sobre el AFF y 256.70 % sobre LP. Aunque, el decremento que presenta el método propuesto en la recuperación en

comparación contra las demás metodologías lo ubica en desventaja en el caso de pretender identificar la mayor cantidad posible de trazas verdaderas. Sin embargo, aún se considera cómo el método preferible dentro del marco de comparación desarrollado en esta tesis, gracias a que mantuvo el mayor puntaje promedio en F2. Esto indica que el método propuesto puede contribuir para reducir la curva de aprendizaje, reducir costo y errores en el desarrollo de proyectos de software. Finalmente, tiene la ventaja de que no es necesario desarrollar una base de datos auxiliar ni tampoco es requerido buscar una tarea de entrenamiento diferente dentro del mismo contexto de la tarea objetivo.

# RECOMENDACIONES

## LIMITACIONES

Algunas limitantes en el presente trabajo se enlistan a continuación:

- **Base de datos:** Todo el trabajo se llevó a cabo solamente en una misma base de datos, por lo tanto, se limitó por falta de pruebas fuera de la distribución, para validar la capacidad del método propuesto para evaluar el modelo generado en términos de generalización de la solución. Sin embargo, se empleó la técnica de doble validación cruzada para evaluar el modelo en un marco de comparación.
- **Etiquetado original de rastreabilidad en la base de datos:** En este trabajo se asume que la identificación de las trazas, elaborada originalmente por los autores de la base de datos empleada en este trabajo, es correcta. Por lo que, cualquier etiquetado incorrecto podría afectar el desempeño final del modelo.
- **Número de experimentos:** La cantidad de experimentos se delimitó a cinco semillas aleatorias por cuestiones de tiempo y recursos, lo que puede ser una limitante en la reproducibilidad de los experimentos, por lo que se complementó la experimentación con doble validación cruzada.

## TRABAJO FUTURO

Algunas opciones de trabajo futuro se enlistan a continuación:

- **Aplicación de la técnica propuesta en otros modelos:** Experimentar con variantes de BERT mediante el método propuesto podría presentar un área de estudio para contribuir en la rastreabilidad de requerimientos. Por ejemplo, se puede aplicar el método propuesto en CodeBERT.
- **Explorar el entrenamiento temprano de otras capas y de parámetros:** Estudiar una selección distinta de capas o parámetros pudiera tener un impacto en el efecto de la técnica propuesta (Ben-Zaken et al., 2021), lo cual podría contribuir con nuevos métodos en el área de estudio.

- **Emplear otras técnicas de reponderación de peso de clases y remuestreo:** Analizar otras alternativas que han demostrado ser efectivas en el manejo de bases de datos desbalanceadas (Cao et al., 2019) podría aportar en el desarrollo y la experimentación en la rastreabilidad de requerimientos de software.
- **Capas de clasificación:** Evaluar el desempeño del mismo método aplicando una capa de clasificación diferente podría contribuir con modelos que presenten un mayor puntaje en F2, por ejemplo, agregando un bloque de clasificación con dos capas.
- **Análisis semántico:** Emplear la salida de BERT para aplicar técnicas de análisis semántico combinado con el ajuste fino empleando el método propuesto podría generar un aporte al área de rastreabilidad de requerimientos.

# APORTACIÓN DE LA TESIS

El estudio desarrollado en esta tesis contribuye al área de ingeniería de requerimientos con un método que propone incrementar el desempeño de modelos de lenguaje en la tarea de rastreabilidad de requerimientos mediante un ajuste fino en dos etapas. En el entrenamiento se seleccionan las capas transformer de BERT según su capacidad de representación semántica y sintáctica, en la primera y segunda etapa respectivamente. Esto ha sido demostrado en un marco de comparación contra tres métodos diferentes en una base de datos de acceso público que contiene requerimientos de software de distintos niveles escritos en lenguaje natural. Además, el método propuesto es una contribución como referencia en el área de transferencia de aprendizaje ya que se propone una estrategia que no había sido empleada antes y que fue sustentado por medio de doble validación cruzada. Adicionalmente, ya que el método propuesto se aplicó en un modelo de lenguaje desarrollado para uso general, puede aplicarse en tareas similares de clasificación de texto. Cabe mencionar que también se publicó el siguiente artículo de investigación: Mendivil, C., y Corral, L. (2019). Revisión literaria concerniente al análisis semántico de requerimientos de un producto de software escritos en lenguaje natural mediante el método de procesamiento de lenguaje natural. Academia Journals, 11(2). https://www.academiajournals.com/pub-morelia-2019.

# APORTACIÓN SOCIAL DE LA TESIS

Esta aplicación es relevante para la industria de desarrollo de software, pues amplía el conocimiento en la ingeniería de software colaborando en reducir la complejidad inherente en la rastreabilidad de requerimientos de software. Particularmente, contribuye en el desarrollo de herramientas para los ingenieros de software en la tarea de análisis de requerimientos de manera complementaria permitiendo la automatización de la rastreabilidad entre requerimientos de distinto nivel de abstracción, pues es capaz de proveer una lista de posibles candidatos de pares de requerimientos donde puede existir una condición de rastreabilidad. Esto se puede resumir en un menor costo de ejecución en los proyectos de software, reducción del impacto en la curva de aprendizaje en esta área y así como en una disminución de los errores encontrados en etapas posteriores de desarrollo.

# REFERENCIAS

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

Ajagbe, M., y Zhao, L. (2022). Retraining a BERT Model for Transfer Learning in Requirements Engineering: A Preliminary Study. 2022 IEEE 30th International Requirements Engineering Conference (RE), 309–315. https://doi.org/10.1109/RE54965.2022.00046

Arora, C., Sabetzadeh, M., Briand, L., y Zimmer, F. (2015). Automated Checking of Conformance to Requirements Templates Using Natural Language Processing. IEEE Transactions on Software Engineering, 41(10), 944–968. https://doi.org/10.1109/TSE.2015.2428709

Bella, E. E., Gervais, M., Bendraou, R., Wouters, L., y Koudri, A. (2018). Semi-Supervised Approach for Recovering Traceability Links in Complex Systems. 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS), 193–196. https://doi.org/10.1109/ICECCS2018.2018.00030

Ben Zaken, E., Goldberg, Y., y Ravfogel, S. (2022). BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. En S. Muresan, P. Nakov, y A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 1–9). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-short.1

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. ArXiv, abs/2005.14165. https://api.semanticscholar.org/CorpusID:218971783

Canedo, E. D., y Mendes, B. C. (2020). Software Requirements Classification Using Machine Learning Algorithms. Entropy, 22. https://api.semanticscholar.org/CorpusID:224820551

Center of Excellence for Software & Systems Traceability (CoEST). (s/f). Recuperado el 1 de septiembre de 2023, de http://sarec.nd.edu/coest/datasets.html

Chawla, N. V. (2005). Data Mining for Imbalanced Datasets: An Overview. En O. Maimon y L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook (pp. 853–867). Springer US. https://doi.org/10.1007/0-387-25465-X_40

Chen, J., Zhang, A., Shi, X., Li, M., Smola, A., y Yang, D. (2023). Parameter-efficient fine-tuning design spaces.

Croft, D., Coupland, S., Shell, J., y Brown, S. (2013). A fast and efficient semantic short text similarity metric. Computational Intelligence (UKCI), 2013 13th UK Workshop on, 221–227.

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv. http://arxiv.org/abs/1810.04805

Fang, C., He, H., Long, Q., y Su, W. J. (2021). Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. Proceedings of the National Academy of Sciences, 118(43). https://doi.org/10.1073/pnas.2103091118

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., y Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. ArXiv, abs/2002.08155. https://api.semanticscholar.org/CorpusID:211171605

Goodfellow, I., Bengio, Y., y Courville, A. (2016). Deep Learning. MIT Press.

Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Yin, J., Jiang, D., y Zhou, M. (2020). GraphCodeBERT: Pre-training Code Representations with Data Flow. ArXiv, abs/2009.08366. https://api.semanticscholar.org/CorpusID:221761146

Guo, H., Zhou, J., y Wu, C.-A. (2018). Imbalanced learning based on data-partition and SMOTE. Information, 9(9), 238.

Guo, J., Cheng, J., y Cleland-Huang, J. (2017). Semantically Enhanced Software Traceability Using Deep Learning Techniques. 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), 3–14. https://doi.org/10.1109/ICSE.2017.9

Haykin, S. (2004). 1 FEEDFORWARD NEURAL NETWORKS : AN INTRODUCTION.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., y Gelly, S. (2019). Parameter-Efficient Transfer Learning for NLP. arXiv. http://arxiv.org/abs/1902.00751

Jawahar, G., Sagot, B., y Seddah, D. (2019). What Does BERT Learn about the Structure of Language? Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 3651–3657. https://doi.org/10.18653/v1/P19-1356

Jayatilleke, S., y Lai, R. (2013). A Method of Specifying and Classifying Requirements Change. 2013 22nd Australian Software Engineering Conference, 175–180. https://doi.org/10.1109/ASWEC.2013.29

Kingma, D. P., y Ba, J. (2017). Adam: A Method for Stochastic Optimization.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., y Liang, P. (2022). Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution.

Laliberte, C. D., Giachetti, R. E., y Kolsch, M. (2022). Evaluation of Natural Language Processing for Requirements Traceability. 2022 17th Annual System of Systems Engineering Conference (SOSE), 21–26. https://doi.org/10.1109/SOSE55472.2022.9812649

Last, F., Douzas, G., y Bacao, F. (2017). Oversampling for Imbalanced Learning Based on K-Means and SMOTE. arXiv:1711.00837 [Cs, Stat]. http://arxiv.org/abs/1711.00837

Lee, J., Tang, R., y Lin, J. (2019). What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning. arXiv. http://arxiv.org/abs/1911.03090

Lemazurier, L., Chapurlat, V., y Grossetete, A. (2017). An MBSE Approach to Pass from Requirements to Functional Architecture. IFAC-PapersOnLine, 50, 7260–7265.

Li, J. J., y Tong, X. (2020). Statistical Hypothesis Testing versus Machine Learning Binary Classification: Distinctions and Guidelines. Patterns, 1(7), 100115. https://doi.org/10.1016/j.patter.2020.100115

Li, X., Gong, Y., Shen, Y., Qiu, X., Zhang, H., Yao, B., Qi, W., Jiang, D., Chen, W., y Duan, N. (2022). CodeRetriever: A Large Scale Contrastive Pre-Training Method for Code Search. Conference on Empirical Methods in Natural Language Processing. https://api.semanticscholar.org/CorpusID:252993162

Li, Y., Yue, T., Ali, S., y Zhang, L. (2019). Enabling automated requirements reuse and configuration. Software & Systems Modeling, 18(3), 2177–2211. https://doi.org/10.1007/s10270-017-0641-6

Lin, J., Liu, Y., Zeng, Q., Jiang, M., y Cleland-Huang, J. (2021). Traceability Transformed: Generating more Accurate Links with Pre-Trained BERT Models. arXiv. http://arxiv.org/abs/2102.04411

Lin, J., Poudel, A., Yu, W., Zeng, Q., Jiang, M., y Cleland-Huang, J. (2022). Enhancing Automated Software Traceability by Transfer Learning from Open-World Data. arXiv. http://arxiv.org/abs/2207.01084

Liu, Y., Agarwal, S., y Venkataraman, S. (2021). AutoFreeze: Automatically Freezing Model Blocks to Accelerate Fine-tuning. arXiv. http://arxiv.org/abs/2102.01386

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., y Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv, abs/1907.11692. https://api.semanticscholar.org/CorpusID:198953378

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., y Brinkkemper, S. (2016). Improving agile requirements: The Quality User Story framework and tool. Requirements Engineering, 21(3), 383–403. https://doi.org/10.1007/s00766-016-0250-x

Madabushi, H. T., Kochkina, E., y Castelle, M. (2020). Cost-Sensitive BERT for Generalisable Sentence Classification with Imbalanced Data. CoRR, abs/2003.11563. https://arxiv.org/abs/2003.11563

Mahmoud, A., y Williams, G. (2016). Detecting, classifying, and tracing non-functional software requirements. Requirements Engineering, 21(3), 357–381. https://doi.org/10.1007/s00766-016-0252-8

Matsuoka, J., y Lepage, Y. (2011). Ambiguity spotting using wordnet semantic similarity in support to recommended practice for software requirements specifications. Natural Language Processing andKnowledge Engineering (NLP-KE), 2011 7th International Conference on, 479–484.

May, S., Hartmann, S., y Klawonn, F. (2022). Combined Pruning for Nested Cross-Validation to Accelerate Automated Hyperparameter Optimization for Embedded Feature Selection in High-Dimensional Data with Very Small Sample Sizes.

McZara, J., Sarkani, S., Holzer, T., y Eveleigh, T. (2015). Software requirements prioritization and selection using linguistic tools and constraint solvers—A controlled experiment. Empirical Software Engineering, 20(6), 1721–1761. https://doi.org/10.1007/s10664-014-9334-8

Miller, G. A. (1994). WordNet: A Lexical Database for English. Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994. https://aclanthology.org/H94-1111

Misra, J. (2016). Terminological inconsistency analysis of natural language requirements. Information and Software Technology, 74, 183–193. https://doi.org/10.1016/j.infsof.2015.11.006

Priya, G. K., y Anupriya, G. (2013). Clustering sentence level-text using fuzzy hierarchical algorithm. Human Computer Interactions (ICHCI), 2013 International Conference on, 1–8.

Radford, A., y Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., y Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

Rago, A., Marcos, C., y Diaz-Pace, J. A. (2016). Assisting requirements analysts to find latent concerns with REAssistant. Automated Software Engineering, 23(2), 219–252. https://doi.org/10.1007/s10515-014-0156-0

Reddi, S. J., Kale, S., y Kumar, S. (2019). On the Convergence of Adam and Beyond.

Rodriguez, A. D., Dearstyne, K. R., y Cleland-Huang, J. (2023a). Prompts Matter: Insights and Strategies for Prompt Engineering in Automated Software Traceability. arXiv. http://arxiv.org/abs/2308.00229

Rodriguez, A. D., Dearstyne, K. R., y Cleland-Huang, J. (2023b). Understanding the Challenges of Deploying Live-Traceability Solutions. arXiv. http://arxiv.org/abs/2306.10972

Sagala, T. W., Wati, T., Budi, N. F. A., y Hidayanto, A. N. (2018). Analysis and Implementation Measurement of Semantic Similarity Using Content Management Information on WordNet. 2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 337–342.

Sanh, V., Debut, L., Chaumond, J., y Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv, abs/1910.01108. https://api.semanticscholar.org/CorpusID:203626972

Shen, Y., y Liu, J. (2021). Comparison of Text Sentiment Analysis based on Bert and Word2vec. 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC), 144–147. https://doi.org/10.1109/ICFTIC54370.2021.9647258

Shi, Y., ValizadehAslani, T., Wang, J., Ren, P., Zhang, Y., Hu, M., Zhao, L., y Liang, H. (2022). Improving Imbalanced Learning by Pre-finetuning with Data Augmentation. En N. Moniz, P. Branco, L. Torgo, N. Japkowicz, M. Wozniak, y S. Wang (Eds.), Proceedings of the Fourth International Workshop on Learning with Imbalanced Domains: Theory and Applications (Vol. 183, pp. 68–82). PMLR. https://proceedings.mlr.press/v183/shi22a.html

Sonbol, R., Rebdawi, G., y Ghneim, N. (2022a). Learning software requirements syntax: An unsupervised approach to recognize templates. Knowl. Based Syst., 248, 108933.

Sonbol, R., Rebdawi, G., y Ghneim, N. (2022b). The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review. IEEE Access, 10, 62811–62830. https://doi.org/10.1109/ACCESS.2022.3182372

Sun, C., Qiu, X., Xu, Y., y Huang, X. (2020). How to Fine-Tune BERT for Text Classification?

Sundaram, S. K., Hayes, J. H., y Dekhtyar, A. (2005). Baselines in Requirements Tracing. Proceedings of the 2005 Workshop on Predictor Models in Software Engineering, 1–6. https://doi.org/10.1145/1082983.1083169

Tian, J., Zhang, L., y Lian, X. (2023). A Cross-Level Requirement Trace Link Update Model Based on Bidirectional Encoder Representations from Transformers. Mathematics, 11(3), 623. https://doi.org/10.3390/math11030623

Tikayat Ray, A., Cole, B. F., Pinon Fischer, O. J., Bhat, A. P., White, R. T., y Mavris, D. N. (2023). Agile Methodology for the Standardization of Engineering Requirements Using Large Language Models. Systems, 11(7), 352. https://doi.org/10.3390/systems11070352

Unterkalmsteiner, M., Gorschek, T., Feldt, R., y Lavesson, N. (2016). Large-scale information retrieval in software engineering—An experience report from industrial application. Empirical Software Engineering, 21(6), 2324–2365. https://doi.org/10.1007/s10664-015-9410-8

ValizadehAslani, T., Shi, Y., Wang, J., Ren, P., Zhang, Y., Hu, M., Zhao, L., y Liang, H. (2022). Two-Stage Fine-Tuning: A Novel Strategy for Learning Class-Imbalanced Data. ArXiv, abs/2207.10858. https://api.semanticscholar.org/CorpusID:251018417

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. (2023). Attention Is All You Need.

Wainer, J., y Cawley, G. (2018). Nested cross-validation when selecting classifiers is overzealous for most practical applications. arXiv. http://arxiv.org/abs/1809.09446

Wang, Y. (2016). Automatic semantic analysis of software requirements through machine learning and ontology approach. Journal of Shanghai Jiaotong University (Science), 21(6), 692–701. https://doi.org/10.1007/s12204-016-1783-3

Wu, D., Huang, J., y Yang, S. (2017). A Joint Model for Sentence Semantic Similarity Learning. 2017 13th International Conference on Semantics, Knowledge and Grids (SKG), 120–125. https://doi.org/10.1109/SKG.2017.00027

Yang, Z., Ding, M., Guo, Y., Lv, Q., y Tang, J. (2022). Parameter-Efficient Tuning Makes a Good Classification Head. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, 7576–7586. https://doi.org/10.18653/v1/2022.emnlp-main.514

Zhao, T., Cao, Q., y Sun, Q. (2017). An Improved Approach to Traceability Recovery Based on Word Embeddings. 2017 24th Asia-Pacific Software Engineering Conference (APSEC), 81–89. https://doi.org/10.1109/APSEC.2017.14

Zheng, Z., Ning, K., Chen, J., Wang, Y., Chen, W., Guo, L., y Wang, W. (2023). Towards an Understanding of Large Language Models in Software Engineering Tasks. arXiv. http://arxiv.org/abs/2308.11396