UNIVERSIDAD NACIONAL DE MOQUEGUA

FACULTAD DE INGENIERÍAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TESIS:

Generación automática de resúmenes de resoluciones judiciales de la Corte Superior de Justicia del Perú mediante técnicas de procesamiento de lenguaje natural

Presentado por:

Bach. Diana Carolina Sucapuca Mamani

Asesor:

Mgr. Hugo Euler Tito Chura

Coasesor:

Ing. Honorio Apaza Alanoca

Para optar el título profesional de :

INGENIERO DE SISTEMAS E INFORMÁTICA

Moquegua, Abril de 2026

UNAM
UNIVERSIDAD NACIONAL DE MOQUEGUA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

Folio **Nº 030**

ACTA DE SUSTENTACIÓN

I. Modalidad:

Tesis : [X]
Trabajo de Suficiencia Profesional : [ ]
Otro: ..... : [ ]

II. Documento de aprobación para la sustentación:

Resolucion de Consejo de Facultad N° 143-2026-CF-EAIN-UNAM

Fecha: 11-05-2026

III. El jurado ha dictaminado la (el) tesis, TITULADO

Generación automática de resúmenes de resoluciones judiciales de la corte superior de justicia del Perú mediante técnicas de procesamiento de lenguaje natural.

Bachiller: Diana Carolina Sucapuca Mamani

Asesor Principal: Hugo Euler Tito Chora

Co-Asesor: Honorio Apaza Alanoca

IV. Se dictamina:

Como: Aprobado por mayoria la sustentación de manera:

Presencial [X] Virtual [ ]

Con nota: trece (13) por lo que; Sí [X] No [ ]

Opta el Título Profesional de INGENIERO DE SISTEMAS E INFORMÁTICA .

Siendo las 14:10 horas, del 14 de mayo del año 2026 se procede con la firma en conformidad de lo dictaminado por el Jurado.

Dr. José A. Guzmán Valdivia
Presidente

Juan Carlos Clares Pérez
Primer Miembro

Dr. Morales Gonzalo Russo
Segundo Miembro

Diana Carolina Sucapuca Mamani
Bachiller

Dedicatoria

A Dios, por haberme dado la fortaleza y la sabiduría necesaria para no desfallecer en los momentos más difíciles de esta investigación.

A mi familia, mi refugio seguro, por su paciencia infinita durante mis largas noches de estudio y por creer en mí incluso cuando yo dudaba. Gracias por enseñarme el valor de la perseverancia y el trabajo honesto.

A mi patria, con la esperanza de que este trabajo contribuya, aunque sea en pequeña medida, a la modernización de nuestra justicia.

Resumen

La presente investigación aborda la problemática de la saturación procesal en el sistema de administración de justicia peruano, proponiendo una solución tecnológica basada en Inteligencia Artificial para la automatización del resumen de resoluciones judiciales. El objetivo principal fue desarrollar y evaluar modelos de aprendizaje profundo basados en la arquitectura Transformer, capaces de generar síntesis coherentes y precisas de sentencias de apelación en materia laboral. Para la experimentación, se construyó un corpus especializado de 6,848 resoluciones judiciales emitidas por la Corte Superior de Justicia de Lima entre los años 2021 y 2022. Se implementó un diseño experimental comparativo evaluando cuatro arquitecturas de vanguardia: mT5, BART, Pegasus y BETO. La evaluación se realizó mediante métricas automáticas estándar (ROUGE y BLEU) y validación cualitativa.

Los resultados demostraron que el modelo mT5 (Multilingual T5), tras un proceso de ajuste fino (fine-tuning), superó significativamente a las otras arquitecturas, alcanzando un coeficiente BLEU de 0.1228 y métricas ROUGE consistentes. El modelo logró abstraer la estructura lógica de las sentencias peruanas, generando resúmenes con alta coherencia sintáctica y fidelidad terminológica. Se concluye que la aplicación de modelos Sequence-to-Sequence es técnica y funcionalmente viable para asistir en la descarga procesal, proporcionando una herramienta eficaz para la gestión documental en el Poder Judicial.

Palabras clave: Procesamiento de Lenguaje Natural, Resumen Automático, Transformers, mT5, Resoluciones Judiciales, Justicia Digital.

Abtract

This research addresses the issue of procedural backlog in the Peruvian justice administration system by proposing an Artificial Intelligence-based solution for the automatic summarization of judicial resolutions. The main objective was to develop and evaluate deep learning models based on the Transformer architecture, capable of generating coherent and accurate syntheses of appellate judgments in labor law.

For the experimentation, a specialized corpus of 6,848 judicial resolutions issued by the Superior Court of Justice of Lima between 2021 and 2022 was constructed. A comparative experimental design was implemented to evaluate four state-of-the-art architectures: mT5, BART, Pegasus, and BETO. Evaluation was conducted using standard automatic metrics (ROUGE and BLEU) and qualitative validation.

The results demonstrated that the mT5 (Multilingual T5) model, following a fine-tuning process, significantly outperformed the other architectures, achieving a BLEU score of 0.1228 and consistent ROUGE metrics. The model successfully abstracted the logical structure of Peruvian sentences, generating summaries with high syntactic coherence and terminological fidelity. It is concluded that the application of Sequence-to-Sequence models is technically and functionally viable to assist in reducing procedural load, providing an effective tool for document management within the Judiciary.

Keywords: Natural Language Processing, Automatic Summarization, Transformers, mT5, Judicial Resolutions, Digital Justice.

Tabla de Contenido

Introducción 12

I. Planteamiento del problema 15

1.1 Descripción y formulación del problema 15

1.2 Objetivo 20

1.2.1 Objetivo General 20

1.2.2 Objetivos Específicos 20

1.3 Justificación e importancia 21

1.4 Variables 22

II. Marco Teórico 24

2.1 Antecedentes 24

2.2 Bases Teóricas 26

2.2.1 Introducción y Contextualización 26

2.2.2 Fundamentos del Procesamiento de Lenguaje Natural 28

2.2.2.1 Procesamiento lenguaje natural: 28

2.2.2.2 Técnicas de Representación 29

2.2.2.3 Sentence Embeddings 30

2.2.3 Modelos de lenguaje 31

2.2.4 BERT (Bidirectional Encoder Representations from Transformers) 31

2.2.4.1 Masked Language Model (MLM) 32

2.2.4.2 Next Sentence Prediction (NSP) 32

2.2.4.3 Estructura BERT 33

2.2.5 BART (Bidirectional and Auto-Regressive Transformers) 34

2.2.5.1 Arquitectura BART 35
2.2.5.2 Preentrenamiento de BART 37
2.2.5.3 Ajuste Fino (Fine Tuning) 38
2.2.5.4 Mecanismos de Atención 39
2.2.6 Técnicas de Resumen Automático 39
2.2.6.1 Métodos tradicionales y estadísticos 40
2.2.6.2 Text Bank 41
2.2.7 Evaluación de Resúmenes automáticos 41
2.2.7.1 Métricas de evaluación 42
2.2.8 Características y desafíos del Lenguaje legal 42
2.3 Definición de términos 44
III. Metodología 46
3.1 Tipo y nivel de investigación 46
3.1.1 Tipo de investigación 46
3.1.2 Nivel de investigación 46
3.2 Ámbito temporal y espacial 46
3.2.1 Ámbito Espacial 47
3.2.2 Ámbito Temático 47
3.2.3 Ámbito Temporal 47
3.3 Población y muestra 47
3.3.1 Población 48
3.3.2 Muestra 48
3.4 Instrumentos 49

3.5 Procedimientos 51
3.5.1 Construcción del Corpus 52
3.5.1.1 Recuperación de documentos legales 53
3.5.1.2 Metodología de extracción automatizada en dos fases 55
3.5.1.3 Exploración de los datos 57
3.5.2 Pre procesamiento 59
3.5.2.1 Limpieza y Normalización del Corpus Legal 60
3.5.2.2 Tokenización y Gestión de Longitud de Secuencia 61
3.5.2.3 Estrategia de División del Conjunto de Datos 65
3.5.3 Modelamiento e implementación 66
3.5.3.1 Selección de Arquitecturas Transformer 67
3.5.3.1.1 Arquitectura Transformer 67
3.5.3.1.2 BART (Bidirectional and Auto-Regressive Transformers): 69
3.5.3.1.3 BETO (BERT en Español – Configuración Encoder-Decoder): 73
3.5.3.1.4 mT5 (Multilingual Text-to-Text Transfer Transformer): 78
3.5.3.1.5 PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization): 84
3.5.3.2 Configuración del Entorno Experimental 89
3.5.3.3 Definición de Hiperparámetros y Ajuste Fino 90
3.5.4 Evaluación de desempeño 93
3.5.4.1 Métricas de Evaluación Automática 93
3.5.4.2 Protocolo de Pruebas con Datos No Vistos 95
3.6 Analisis de datos 95

3.6.1 Métodos de análisis de datos 96
3.6.2 Técnicas para la presentación de datos 97
IV. Resultados y discusion 99
4.1 Resultados 99
4.1.1 Dinámica de aprendizaje y validación del ajuste 99
4.1.1.1 Evaluación comparativa de la convergencia de la función de pérdida 100
4.1.1.2 Análisis de estabilidad y generalización por arquitectura 102
4.1.1.3 Comportamiento de generación y métricas intermedias 106
4.1.2 Evaluación del desempeño sobre el conjunto de prueba (test set) 108
4.1.2.1 Comparativa de métricas de generación automática (ROUGE y BLEU) 108
4.1.2.2 Análisis cuantitativo de la fidelidad textual 109
4.1.2.3 Análisis Cualitativo 112
4.2 Discusión de resultados 115
4.2.1 La paradoja de la pérdida vs. calidad semántica 115
4.2.2 Limitaciones estructurales de los modelos Encoder 116
4.2.3 Eficacia de la transferencia de aprendizaje multilingüe 116
V. Conclusiones 118
5.1 Sobre la construcción del corpus jurídico 118
5.2 Sobre el procesamiento del lenguaje natural 118
5.3 Sobre la comparación de modelos abstractivos 119
5.4 Sobre la evaluación de la calidad de los resúmenes 119
5.5 Conclusión General 119
VI. Recomendaciones 120

VII. Referencias Bibliográficas 122

VIII. Anexos 127

Anexo A: Códigos QR para acceder al drive de código fuente 127

Anexo B. Matriz de consistencia 128

Lista de Figuras

Figura 1 Ejemplo de Tokenización de palabras 28

Figura 2 Arquitectura BERT 31

Figura 3 Comparación entre BART con BERT y GPT 34

Figura 4 Arquitectura Transformers 36

Figura 5 Metodología 52

Figura 6 Interfaz de la plataforma de Jurisprudencia Nacional Sistematizada, fuente primaria de recolección 55

Figura 7 Vista preliminar del dataframe con el corpus de jurisprudencias procesado. 58

Figura 8 Distribución de frecuencias de las materias legales en el corpus 59

Figura 9 Distribución de frecuencia de la longitud de tokens (estimada) en el corpus de resoluciones judiciales. (Aquí insertas tu gráfico del histograma). 61

Figura 10 Esquema del algoritmo BPE, ilustrando la transición de una representación basada en caracteres a una basada en sub-palabras mediante la fusión de pares frecuentes. 63

Figura 11 Esquema del algoritmo de tokenización WordPiece (Case Sensitive). 64

Figura 12 Arquitectura original del Transformer propuesta por Vaswani et al. (2017). 68

Figura 13 Evolución comparativa de la pérdida de validación (validation loss) durante el ajuste fino para los modelos BART, BETO, mT5 y Pegasus. 101

Figura 14 Dinámica de aprendizaje detallada para el modelo BART: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss). 103

Figura 15 Dinámica de aprendizaje detallada para el modelo BETO: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss). 104

Figura 16 Dinámica de aprendizaje detallada para el modelo mT5: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss). 105

Figura 17 Dinámica de aprendizaje detallada para el modelo Pegasus: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss). 106

Figura 18 Resultados finales de la evaluación automática sobre el conjunto de prueba. 109

Lista de Tablas

Tabla 1 Operacionalización de variables 23

Tabla 2 Distribución de la muestra por subconjuntos de datos 49

Tabla 3 Métricas de validación en la última época del entrenamiento. 107

Tabla 4 Resultados finales de la evaluación automática sobre el conjunto de prueba. 110

Tabla 5 Análisis de Precisión Extractiva (Exp. N° 05044-2020). 113

Introducción

En la última década, la administración de justicia a nivel global ha experimentado una presión sin precedentes debido al incremento exponencial de la carga procesal. En el contexto peruano, esta problemática es particularmente crítica en las salas laborales, donde la acumulación de expedientes genera retrasos significativos en la emisión de fallos y en el acceso ciudadano a una justicia oportuna. Ante este escenario, la digitalización de expedientes iniciada por el Poder Judicial ha abierto una oportunidad única para la aplicación de tecnologías de vanguardia como la Inteligencia Artificial (IA) y el Procesamiento de Lenguaje Natural (PLN).

La presente investigación se centró en la problemática de la saturación informativa que enfrentan los magistrados y auxiliares jurisdiccionales de la Corte Superior de Justicia de Lima. La lectura y análisis manual de extensas sentencias de apelación no solo consume tiempo valioso, en su lugar aumenta el riesgo de errores en la identificación de los puntos resolutivos clave. Por ello, el objetivo principal de este trabajo fue desarrollar y evaluar modelos de aprendizaje profundo basados en la arquitectura Transformer para la generación automática de resúmenes abstractivos de resoluciones judiciales.

A diferencia de los métodos tradicionales de resumen extractivo, que se limitan a seleccionar oraciones literales, esta investigación apostó por un enfoque abstractivo. Se emplearon modelos de última generación como mT5, BART y Pegasus, los cuales fueron sometidos a un proceso de ajuste fino (fine-tuning) utilizando un corpus especializado de sentencias laborales peruanas construido específicamente para este propósito. Asimismo, se implementó la estrategia técnica Head-Tail para superar las limitaciones de ventana de contexto de los modelos actuales, permitiendo procesar documentos extensos sin perder la coherencia jurídica.

La evaluación de los resultados se realizó mediante métricas estándar de la industria como ROUGE y BLEU, complementadas con un análisis cualitativo de la fidelidad fáctica. Los hallazgos demostraron que el modelo mT5 alcanzó el desempeño más equilibrado, logrando sintetizar sentencias complejas en textos concisos y comprensibles, sentando así un precedente tecnológico para la modernización del sistema judicial nacional.

Finalmente, este documento se encuentra estructurado en seis capítulos que detallan el proceso de investigación, desde la fundamentación teórica hasta la validación experimental:

- Capítulo I: Planteamiento del Problema. En esta sección se describe la problemática de la carga procesal en el sistema judicial peruano y se definen los objetivos, la justificación y los alcances de la investigación orientada a la generación de resúmenes.
- Capítulo II: Marco Teórico. Comprende la revisión de antecedentes nacionales e internacionales, así como las bases teóricas de la arquitectura Transformer, los mecanismos de atención y los modelos de lenguaje específicos empleados en el estudio.
- Capítulo III: Marco Metodológico. Detalla el diseño de investigación experimental, la operacionalización de variables y la metodología de construcción del corpus de 6,848 resoluciones judiciales, incluyendo las fases de limpieza y preprocesamiento.
- Capítulo IV: Desarrollo de la Solución. Describe la implementación técnica del entorno experimental, el proceso de fine-tuning de los modelos mT5, BART, Pegasus y BETO, y la aplicación de la estrategia Head-Tail.
- Capítulo V: Resultados y Discusión. Presenta el análisis comparativo de los modelos mediante métricas cuantitativas (ROUGE y BLEU) y una evaluación cualitativa de los resultados obtenidos frente a las hipótesis planteadas.

- Capítulo VI: Conclusiones y Recomendaciones. Sintetiza los hallazgos más relevantes del estudio y propone líneas de investigación futuras para la escalabilidad de soluciones de IA en el ámbito jurisdiccional.

I. Planteamiento del problema

1.1 Descripción y formulación del problema

La transformación digital a nivel mundial ha generado un volumen sin precedentes de documentos legales en formato electrónico, un fenómeno que se ha visto acentuado por la adopción de sistemas como el Expediente Judicial Electrónico (EJE) en el Perú. Este escenario ha impulsado el desarrollo de técnicas avanzadas de Procesamiento de Lenguaje Natural (PLN) para la generación automática de resúmenes, con el objetivo de facilitar el acceso a la información y optimizar el análisis de textos complejos. No obstante, esta abundancia de datos no solo representa una oportunidad, en su lugar también plantea importantes desafíos tanto sociales como técnicos, los cuales impactan directamente en la eficiencia del sistema judicial y en el acceso a la justicia.

Desde una perspectiva internacional, la literatura académica ha explorado ampliamente la complejidad inherente a los textos legales, caracterizados por estructuras intrincadas, terminología especializada, alta ambigüedad y argumentación extensa. Autores como Duong et al. (2023) demuestran la capacidad de sistemas basados en aprendizaje profundo para generar resúmenes concisos de documentos judiciales, incluso capturando la semántica a través de embeddings sin depender de características manuales o conocimiento de dominio específico. De igual forma, Mullick et al. (2022) han desarrollado marcos de evaluación para la generación automática de resúmenes legales que incorporan métricas basadas en la intención, subrayando la importancia de medir la fidelidad y relevancia de los resúmenes en función de sus componentes críticos, como las partes, los hechos, las decisiones o las leyes citadas. La relevancia de estas investigaciones es innegable, ya que sientan las bases metodológicas y técnicas para la aplicación de PLN en contextos jurídicos.

En otra línea, Tran et al. (2023) introducen la técnica de encoded summarization, la cual transforma los documentos legales en representaciones vectoriales continuas, integrando tanto características léxicas como latentes para mejorar la recuperación de casos legales. Además, la encuesta de Reddy & Jayabharathy (2024) analiza el impacto de modelos basados en transformers y arquitecturas seq2seq en la tarea de resumir textos jurídicos, evidenciando el potencial de estas metodologías a escala global, con una particular mención a la mejora en la calidad y coherencia de los resúmenes abstractivos generados. Estos avances son cruciales, ya que los modelos de deep learning, especialmente los transformers, han demostrado una capacidad superior para manejar las sutilezas del lenguaje y generar textos coherentes.No obstante, la aplicación directa de estos métodos en contextos locales, como el sistema judicial peruano, presenta desafíos adicionales que requieren una investigación y adaptación específica. La diversidad en la redacción de resoluciones judiciales, influenciada por la particularidad de cada caso, el estilo de cada magistrado y la evolución de la jurisprudencia, junto con la escasez crítica de conjuntos de datos etiquetados específicos para el derecho peruano, dificultan significativamente la implementación exitosa y el rendimiento óptimo de modelos desarrollados en otros países. Como señala un informe del Poder Judicial del Perú, la implementación del EJE desde 2017 busca modernizar el sistema y digitalizar expedientes, pero la creación de valor a partir de esa digitalización, como la sumarización automática, aún enfrenta barreras importantes. En este contexto, el enfoque de Duong, Nguyen, Vuong, Le, & Nguyen (2023), que utiliza técnicas de deep clustering para ajustar la puntuación de relevancia de las oraciones, resulta prometedor para adaptar estos sistemas al entorno peruano, al permitir una mejor ponderación de la información clave, mejorando así la precisión del análisis documental y, en consecuencia, facilitando el acceso a la justicia.

La falta de corpus de datos legales anotados o etiquetados es una barrera fundamental no únicamente en nuestro país, sino en gran parte de América Latina. La importancia de los corpus en PLN es innegable, ya que son esenciales para que las máquinas aprendan y entiendan el lenguaje, y su construcción es un proceso meticuloso que requiere planificación y anotación experta. Sin estos recursos, la capacidad de entrenar y validar modelos de PLN con la especificidad lingüística y jurídica del Perú se ve severamente limitada, obligando a una dependencia de modelos preentrenados en otros idiomas o dominios, lo que inevitablemente reduce su precisión en el contexto legal peruano.

Por otro lado, a nivel técnico, a pesar de los avances en deep learning, se ha evidenciado que la sobredependencia en resúmenes generados automáticamente puede derivar en omisiones de matices críticos o errores en la síntesis, como advierte Wired (2024). Estos riesgos son particularmente elevados en el dominio legal, donde la precisión y la fidelidad a la información original son fundamentales. Un error en un resumen puede tener consecuencias jurídicas significativas, afectando la toma de decisiones y el proceso judicial. Esto resalta la necesidad de continuar investigando soluciones robustas y contextualizadas que combinen estrategias extractivas y abstractivas, garantizando que los resúmenes sean precisos, coherentes y adaptados a las particularidades del lenguaje jurídico peruano, integrando no solo la literalidad, sino también la capacidad de comprensión y síntesis (como en los abstractivos), siempre con un control de la factualidad.

Adicionalmente, la complejidad del lenguaje jurídico en español presenta matices que a menudo son subestimados por modelos entrenados predominantemente en inglés. El español legal se caracteriza por una mayor flexibilidad sintáctica, el uso frecuente de construcciones pasivas y un vocabulario altamente especializado que varía incluso entre jurisdicciones de habla hispana

(Griselda Areli Matías-Mendoza Yulia Ledeneva, n.d.). Esta variabilidad, sumada a la idiosincrasia de la redacción jurídica peruana, que puede incluir arcaísmos, latinismos y referencias intertextuales implícitas, exige que los modelos de PLN sean capaces de una comprensión semántica profunda que va más allá de la mera detección de palabras clave. La creación de modelos lingüísticos específicos para el español legal peruano, como se ha intentado con éxito en otros dominios específicos del español, se vuelve indispensable para asegurar que la inteligencia artificial pueda interpretar con la precisión requerida los documentos judiciales, evitando así omisiones o interpretaciones erróneas que tendrían graves consecuencias.

Finalmente, el desafío no se circunscribe únicamente a la construcción de los modelos, sino que comprende igualmente su evaluación y su adopción por parte de los usuarios finales del sector legal. La evaluación de la calidad de los resúmenes en el ámbito jurídico es intrínsecamente compleja, ya que no existe un resumen perfecto único y la relevancia de la información puede variar según el propósito del usuario (por ejemplo: un juez, un abogado litigante, un estudiante de derecho) (Mullick et al., 2022). Esto implica no solo emplear métricas automáticas como ROUGE y BLEU, sino también incorporar evaluaciones cualitativas realizadas por especialistas en el ámbito jurídico, quienes puedan verificar la fidelidad, coherencia y utilidad práctica de los resúmenes producidos. Superar la resistencia a la adopción tecnológica en un sector tradicionalmente conservador, como el legal en América Latina, dependerá en gran medida de la capacidad de los sistemas de PLN para demostrar una fiabilidad y un valor tangible, que solo se lograrán mediante la adaptación meticulosa y la validación rigurosa en el contexto local.

El actual método de resumen manual, además de ser costoso en términos de tiempo y recursos humanos, impone una carga considerable sobre el personal judicial. Abogados y jueces dedican innumerables horas a la lectura exhaustiva y la síntesis de documentos extensos, lo que desvía su

atención de tareas de mayor valor estratégico y procesal. Esta ineficiencia va más allá de la ralentización de los procesos judiciales, impactando negativamente en la celeridad que demanda la administración de justicia, de igual manera puede conducir a la fatiga documental y, consecuentemente, a errores o pasajes importantes que podrían pasarse por alto en resoluciones críticas. Un estudio sobre la gestión del conocimiento en sistemas judiciales de América Latina resalta la necesidad urgente de herramientas que optimicen el manejo de grandes volúmenes de información para evitar cuellos de botella y mejorar la productividad operativa.

La sobrecarga de información en el ámbito judicial, exacerbada por la proliferación de expedientes electrónicos, no se limita a afectar la eficiencia operativa; asimismo, puede comprometer la calidad de las decisiones judiciales. Cuando los profesionales del derecho se enfrentan a un volumen inmanejable de texto, la capacidad de identificar patrones, precedentes relevantes o inconsistencias lógicas se ve mermada. Esto va más allá de las implicaciones para la equidad y la coherencia jurisprudencial, en su lugar también incrementa el riesgo de denegación de justicia o de resoluciones que no consideran todos los elementos esenciales de un caso complejo (Ramírez & Soto, 2022). La implementación de un sistema de resumen automático, validado y fiable, no se limitaría a liberar al personal de tareas repetitivas, en su lugar también les proporcionaría una herramienta invaluable para navegar esta vasta cantidad de información de manera más efectiva, permitiéndoles concentrarse en el razonamiento jurídico y la aplicación de la ley.

1.2 Objetivo

1.2.1 Objetivo General

Desarrollar un modelo de IA para generar resúmenes automáticos de resoluciones judiciales emitidas por la Corte Superior de Lima, Perú.

1.2.2 Objetivos Específicos

- Diseñar y construir un corpus de resoluciones judiciales de la Corte Superior de Lima, Perú, etiquetado con resúmenes de referencia, para su uso en el entrenamiento y evaluación de modelos.
- Describir y aplicar técnicas de procesamiento de lenguaje natural orientadas a la segmentación y limpieza de los textos judiciales seleccionados.
- Evaluar el desempeño de modelos abstractivos en la generación automática de resúmenes de resoluciones judiciales.
- Evaluar la calidad de los resúmenes generados a partir de resoluciones judiciales de la Corte Superior de Lima, Perú utilizando métricas como ROUGE y BLEU.

1.3 Justificación e importancia

El volumen creciente de resoluciones judiciales emitidas por la Corte Superior de Lima, Perú dificulta su consulta y análisis eficiente. Estos documentos, que contienen información crucial para jueces, abogados y ciudadanos, presentan una extensión y complejidad que obstaculizan la identificación rápida de elementos clave. Actualmente, los resúmenes de resoluciones son generados de manera manual, lo que implica un proceso costoso en términos de tiempo y recursos humanos, creando un cuello de botella en el flujo de información judicial. La falta de mecanismos automatizados para la síntesis de estos textos representa un desafío significativo para la eficiencia del sistema judicial y el acceso equitativo a la información legal.

El procesamiento de lenguaje natural ha demostrado ser una herramienta poderosa en la automatización de tareas relacionadas con el análisis de textos jurídicos a nivel global. En particular, la generación automática de resúmenes puede transformar la forma en que se accede y se comprende la información judicial, al proporcionar versiones condensadas que resalten los aspectos más relevantes de cada caso. No obstante, la aplicación de modelos abstractivos en este contexto, y especialmente en el ámbito judicial peruano, sigue siendo un área de estudio en desarrollo, con desafíos específicos relacionados con la naturaleza del lenguaje jurídico local y la disponibilidad de datos.

Dado este escenario, la presente investigación busca contribuir de manera fundamental al desarrollo de soluciones basadas en PLN para el sector legal peruano. Debido a que a pesar de los avances internacionales, especialmente en países hispanohablantes como Perú, la aplicación de modelos de resumen automático en contextos legales locales es limitada. Al integrar evidencia empírica sobre el rendimiento de los modelos de extracción y abstractos en textos legales

peruanos, este estudio proporciona un marco conceptual para futuras investigaciones en el campo de la tecnología jurídica.

Este estudio mediante pruebas y comparaciones, permitiódesarrollar un modelo de inteligencia artificial e identificar directamente datos en tiempo real del sistema judicial peruano. La creación de un corpus de textos especializado, etiquetado con resúmenes de referencia, fue un elemento clave que permitió una evaluación rigurosa de los modelos utilizando métricas objetivas como ROUGE y BLEU. Esta estrategia metodológica se centra en el valor y la precisión de los enfoques abstractos, sentando un precedente para la investigación reproducible en el ámbito jurídico.

Los resultados de este estudio tendrán un impacto significativo en la optimización del acceso a la información jurídica, reduciendo el tiempo necesario para el análisis de documentos legales por parte de profesionales y ciudadanos. Además, la implementación de un sistema automatizado de resúmenes puede mejorar la transparencia en el sistema judicial al facilitar la difusión de información clave a un público más amplio, lo que es vital para la rendición de cuentas y la legitimidad de las decisiones judiciales. En este sentido, la investigación trasciende la relevancia académica, al contribuir al estado del arte en PLN y LegalTech en español, en su lugar también aporta beneficios prácticos y tangibles al sistema de justicia y a la sociedad en general, alineándose con los objetivos de modernización y acceso a la justicia que impulsa el Poder Judicial del Perú.

1.4 Variables

Para esta investigación se han identificado dos variables principales: la **variable independiente**, correspondiente al tipo de modelo de generación de resúmenes abstractivos; y la **variable dependiente**, que representa la calidad del resumen generado. Esta calidad fue evaluada

mediante métricas cuantitativas objetivas, tales como ROUGE y BLEU, que permiten comparar el contenido generado con resúmenes de referencia.

Tabla 1

Operacionalización de variables

Variable	Definición	Definicion operacional	Dimensiones	Indicadores
Independiente: Modelos de generación automática de resúmenes.	Sistemas de IA basados en modelos de lenguaje para la producción de texto sintetizado (Vaswani et al., 2017).	Implementación y fine-tuning de arquitecturas Transformer (mT5, BART, Pegasus, BETO) sobre el corpus judicial.	Técnicas de Procesamiento de Lenguaje Natural (PLN).	• Modelo BART • Modelo T5 • Modelo PEGASUS • Modelo Transformer
Dependiente: Calidad del resumen generado.	Nivel de precisión, fluidez y cobertura informativa de la síntesis frente a un patrón de referencia (Lin, 2004).	Medición cuantitativa mediante algoritmos de comparación de n-gramas entre la salida del modelo y el gold standard.	Coherencia y similitud textual (Papineni et al., 2002).	• ROUGE-1 • ROUGE-2 • ROUGE-L • BLEU score

Nota. Elaboración propia.

Las métricas ROUGE y BLEU permitió evaluar la similitud entre los resúmenes generados y los resúmenes de referencia. De este modo, se podrá determinar qué enfoque de generación de resúmenes ofrece mejor desempeño en el contexto de resoluciones judiciales de la Corte Superior de Lima, Perú.

II. Marco Teórico

2.1 Antecedentes

La digitalización de documentos jurídicos ha generado un crecimiento exponencial en la cantidad de resoluciones judiciales, lo que plantea desafíos en la eficiencia y accesibilidad de la información dentro del sistema judicial. En el caso de la Corte Superior de Lima, Perú, la ausencia de mecanismos automatizados para resumir y organizar estos textos dificulta el acceso a información clave y ralentiza los procesos de consulta y análisis de jurisprudencia.

La generación automática de resúmenes de textos legales ha sido un área de investigación creciente debido a la necesidad de manejar grandes volúmenes de documentos jurídicos. Los métodos tradicionales de resumen se han basado en enfoques extractivos y abstractivos, cada uno con ventajas y desafíos particulares. Anand & Wagh (2022) propusieron un enfoque basado en redes neuronales para la tarea de resumen automático de documentos legales en la India. Su modelo emplea embeddings de palabras y oraciones para capturar la semántica sin depender de características diseñadas manualmente o conocimientos específicos del dominio.

Otros estudios han explorado diversas estrategias de aprendizaje profundo para la tarea de resumen legal. Sheik & Nirmala (2021) realizaron un análisis comparativo de modelos secuencia a secuencia y técnicas de transferencia de aprendizaje, destacando la efectividad de modelos preentrenados para mejorar la calidad de los resúmenes generados. El uso del análisis semántico latente (LSA) en la tarea de resumen legal ha sido abordado por Merchant & Pande (2018), quienes implementaron un sistema para generar resúmenes de documentos judiciales basado en la extracción de conceptos dentro de un solo documento y en múltiples documentos.

Asimismo, Polsley et al. (2016) desarrollaron **CaseSummarizer**, una herramienta de resumen automático que combina métodos basados en frecuencia de palabras con conocimiento específico del dominio legal, proporcionando resúmenes con mapas de calor y controles flexibles.

El resumen automático en lenguas menos representadas, como el griego, ha sido explorado por (Koniaris et al., 2023), quienes desarrollaron un nuevo conjunto de datos y evaluaron varios métodos de resumen extractivo y abstráctivo en documentos legales de Grecia, resaltando la necesidad de métricas más precisas para evaluar la coherencia y relevancia de los resúmenes. Por otro lado, (Dixit et al., 2024) analizaron el impacto de las técnicas extractivas en la generación de resúmenes legales, destacando que los métodos extractivos preservan mejor la estructura y el significado de los documentos originales en comparación con los métodos abstractivos.

El uso de arquitecturas basadas en seq2seq para la generación de resúmenes ha sido explorado por (Prasad et al., 2020), quienes aplicaron redes neuronales recurrentes (RNN) para generar resúmenes abstractivos con puntuaciones ROUGE entre 0.6 y 0.7. Una visión general sobre las técnicas de resumen judicial fue presentada por (Zhao, 2024), quienes compararon diferentes enfoques extractivos y generativos, además de analizar el impacto del aprendizaje por transferencia en la generación de resúmenes de textos judiciales.

Finalmente, Chen et al. (2024) propusieron una mejora del modelo LSTM para la tarea de resumen legal, integrando información temática y artículos de ley en el proceso de clasificación de oraciones, logrando mejorar la extracción de características en textos legales extensos.

A pesar de estos avances, la aplicación de estas tecnologías en el contexto peruano sigue siendo limitada. La necesidad de adaptar los modelos de PLN a las particularidades del lenguaje jurídico utilizado en el Perú, lo que representa un reto clave para la optimización de estos sistemas.

Desarrollar herramientas que automaticen el procesamiento de documentos legales en español es urgente, dado el alto volumen de resoluciones judiciales que deben ser analizadas diariamente. Estos antecedentes justifican la necesidad de desarrollar un modelo de generación automática de resúmenes específicamente diseñado para las resoluciones judiciales de la Corte Superior de Lima, Perú, con el objetivo de mejorar la accesibilidad, eficiencia y precisión en la consulta de información legal.

2.2 Bases Teóricas

2.2.1 Introducción y Contextualización

La transformación digital ha revolucionado diversos sectores, y el ámbito jurídico no es la excepción. La creciente digitalización de documentos legales ha permitido que grandes volúmenes de información previamente en papel se encuentren disponibles en formato electrónico, facilitando el acceso y la consulta de resoluciones, sentencias y demás textos jurídicos, tal es el caso Jurisprudencia Nacional Sistematizada. Esta evolución no solo permite la modernización de los sistemas de administración de justicia, en su lugar también ha generado nuevos desafíos, como la sobrecarga informativa y la necesidad de procesar y analizar textos extensos y complejos de manera eficiente.

En este contexto, según el estado de arte el Procesamiento de Lenguaje Natural (PLN) se ha posicionado como una herramienta esencial para automatizar tareas de análisis y resumen de documentos legales. Los avances en técnicas de PLN han permitido desarrollar modelos capaces de extraer y condensar la información relevante, lo que resulta crucial para la toma de decisiones por parte de abogados, jueces y otros operadores del derecho. Por ejemplo, Manning & Schütze (1999) sentaron las bases para el análisis estadístico del lenguaje, lo que ha permitido

posteriormente la creación de técnicas de representación de textos (como word embeddings) que facilitan el procesamiento automático de grandes volúmenes de información.

Asimismo, la digitalización ha transformado la forma en que se administran y consultan los documentos judiciales. En la investigación de Susskind (2013) destaca en su obra Tomorrow's Lawyers que la evolución tecnológica no solo mejora la eficiencia en el acceso a la información legal, en su lugar también redefine las competencias y el rol de los profesionales del derecho en la era digital. En línea con esta perspectiva, Asi mismo en la investigación de Ashley (2017) argumenta que la inteligencia artificial y el análisis legal mediante técnicas avanzadas ofrecen nuevas herramientas para la práctica jurídica, permitiendo un análisis más rápido y preciso de documentos complejos.

La importancia de automatizar la generación de resúmenes radica en la necesidad de mitigar el problema de la sobrecarga informativa. La enorme cantidad de documentos disponibles en formato digital requiere herramientas que permitan extraer rápidamente la información esencial, optimizando así la consulta judicial y reduciendo el tiempo que los profesionales deben dedicar a tareas repetitivas. Esta problemática es especialmente relevante en sistemas judiciales donde la consulta y análisis manual de los documentos resulta inviable dada la magnitud de la información disponible.

La digitalización de documentos legales y el desarrollo de técnicas de PLN han creado un entorno en el que la automatización de la generación de resúmenes se convierte en una necesidad imperante. La revisión de la literatura respalda que estas tecnologías tienen el potencial de transformar la administración de justicia, mejorando la eficiencia y facilitando el acceso a la información, lo cual es fundamental para la toma de decisiones en el ámbito jurídico.

2.2.2 Fundamentos del Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural es un campo interdisciplinario que se sitúa en la intersección de la lingüística, la informática y la inteligencia artificial. Su objetivo es dotar a las computadoras de la capacidad de interpretar, analizar y generar lenguaje humano de manera que pueda ser utilizado en una amplia variedad de aplicaciones, desde la traducción automática hasta la extracción de información y la generación de resúmenes (Jurafsky & Martin, 2018).

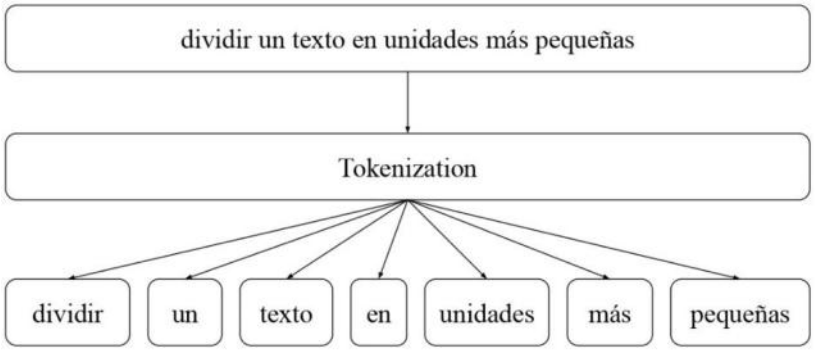
2.2.2.1 Procesamiento lenguaje natural:

El PLN abarca tareas fundamentales como:

- **Tokenización:** Proceso fundamental en el PLN que consiste en dividir un texto en unidades más pequeñas llamadas tokens. Estos tokens pueden ser palabras individuales, signos de puntuación, números o incluso caracteres individuales. La tokenización facilita el análisis posterior del texto, como el conteo de palabras, la identificación de frases clave o la creación de modelos de lenguaje.

  Figura 1

  Ejemplo de tokenización de palabras



  Nota. Elaboración propia

- **Lematización y Derivación:** Técnicas utilizadas para reducir las palabras a su forma base o raíz. La lematización se enfoca en encontrar el lema de una palabra, que es su

forma canónica que aparece en el diccionario. Por ejemplo, corriendo se lematizaría como correr. La derivación, por otro lado, se centra en eliminar prefijos y sufijos para obtener la raíz de una palabra. Por ejemplo, desarrollando se derivaría a desarrollo. Ambas técnicas son útiles para normalizar el texto y facilitar la comparación de palabras.

- **Análisis Sintáctico y Semántico:** Métodos que permiten comprender la estructura y el significado del lenguaje. El análisis sintáctico se encarga de identificar la estructura gramatical de las oraciones, como cuáles son los sujetos, verbos y objetos. Esto ayuda a entender cómo se organizan las palabras para crear significado. El análisis semántico, por su parte, se enfoca en extraer el significado subyacente de las oraciones, incluyendo las relaciones entre las palabras y los conceptos que representan. Ambos tipos de análisis son esenciales para tareas como la traducción automática, la comprensión de lectura y la respuesta a preguntas.

2.2.2.2 Técnicas de Representación

La representación de texto es crucial en el PLN, ya que permite convertir datos textuales en vectores numéricos que capturan sus propiedades semánticas y sintácticas. Entre las técnicas más utilizadas se encuentran:

- **Word Embeddings:** Los *word embeddings* son representaciones vectoriales de palabras que han revolucionado la forma en que las computadoras entienden y utilizan el lenguaje. La idea principal es que palabras con significados similares o que se usan en contextos parecidos tengan vectores cercanos en un espacio multidimensional. Esto permite a los modelos de lenguaje capturar relaciones semánticas sutiles, como sinónimos, analogías e incluso relaciones entre conceptos. A continuación algunos ejemplos de aplicación:

- **Word2Vec:** Desarrollado por Google, Word2Vec utiliza redes neuronales para aprender estos vectores (Mikolov, Chen, Corrado, y Dean, 2013). Tiene dos arquitecturas principales:
    - ***CBOW (Continuous Bag of Words):*** Predice una palabra basándose en las palabras de su contexto.
    - ***Skip-Gram:*** Predice las palabras del contexto a partir de una palabra dada.
- **GloVe.** Creado en Stanford, GloVe se basa en la matriz de co ocurrencia global de palabras en un corpus (Devlin et al., 2018). Combina las ventajas de los métodos globales de factorización de matrices y los métodos locales basados en ventanas.

2.2.2.3 Sentence Embeddings

Los sentence embeddings van un paso más allá de las palabras individuales y generan vectores que representan oraciones completas. Esto es crucial para tareas donde el significado de una oración en su totalidad es importante. A continuación, algunas técnicas destacadas:

- **Sentence BERT:** Extiende BERT para producir embeddings de oraciones de alta calidad. Es particularmente útil para tareas de similitud semántica (Reimers & Gurevych, 2019)(Cer et al., 2018). En la siguiente subsección se desarrolla a mayor detalle.
- **Universal Sentence Encoder:** Desarrollado por Google, este modelo está entrenado en múltiples tareas y produce embeddings genéricos que funcionan bien en diversas aplicaciones (Cer et al., 2018).

  Algunas posibles aplicaciones de estas técnicas son:
    - Similitud de oraciones: Determinar si dos oraciones son semánticamente similares.
    - Clasificación de texto: Asignar categorías a oraciones o documentos.
    - Recuperación de información: Encontrar oraciones relevantes para una consulta dada.

- Parafraseo: Identificar si dos oraciones expresan la misma idea con diferentes palabras.

2.2.3 Modelos de lenguaje

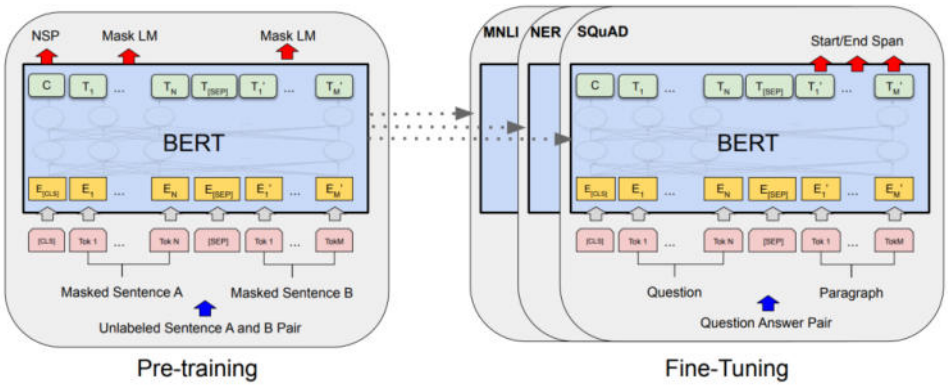
Una vez que se han obtenido representaciones numéricas del lenguaje, se pueden construir modelos de lenguaje que predicen la probabilidad de ocurrencia de secuencias de palabras y realizan tareas de generación de texto. Entre los modelos preentrenados más influyentes se encuentran: son BERT, GPT Y BART, las cuales se desarrollan mas detalladamente en la siguientes secciones.

2.2.4 BERT (Bidirectional Encoder Representations from Transformers)

Es un modelo basado en la arquitectura Transformer que se caracteriza por aprender representaciones contextuales profundas del lenguaje mediante un enfoque bidireccional. A diferencia de modelos previos que procesaban el texto de manera unidireccional (de izquierda a derecha o viceversa), BERT emplea un mecanismo de entrenamiento denominado *Masked Language Model (MLM),* que permite al modelo aprender información contextual de ambos lados de una palabra dentro de una oración (Devlin et al., 2018)..

Figura 2

Arquitectura BERT



Nota. El modelo usa enmascaramiento de palabras para aprender representaciones contextuales (Devlin et al., 2018).

2.2.4.1 Masked Language Model (MLM)

Para entrenar BERT de manera bidireccional, se usa el modelo de lenguaje enmascarado *Masked Language Model, MLM*. En este enfoque, se seleccionan aleatoriamente palabras dentro del texto de entrada y se reemplazan con un token especial **[MASK]**. El modelo debe predecir la palabra correcta basándose en el contexto de las palabras circundantes.

Matemáticamente, el objetivo de MLM se expresa como:

$$P(w_i \mid w_1, \ldots, w_{i-1}, [\boldsymbol{MASK}], w_{i+1}, \ldots, w_N)$$

Donde:

- (w_i) es la palabra enmascarada.
- $(P(w_i))$ es la probabilidad de que la palabra correcta ocupe esa posición.
- El modelo utiliza las palabras anteriores y posteriores a (w_i) para realizar la predicción.

Este proceso obliga a BERT a aprender representaciones más completas del lenguaje, ya que no puede basarse únicamente en palabras precedentes o siguientes, en su lugar debe capturar dependencias bidireccionales.

2.2.4.2 Next Sentence Prediction (NSP)

BERT también se entrena en la tarea de predicción de la siguiente oración *Next Sentence Prediction, NSP*, lo que le permite comprender relaciones entre frases. En este método, al modelo se le proporcionan dos oraciones y debe predecir si la segunda sigue naturalmente a la primera o si es una oración aleatoria.

Matemáticamente, la predicción de la siguiente oración se modela como un problema de clasificación binaria:

$$P(y \mid S_1, S_2) = \text{softmax}(W[S_1; S_2] + b)$$

Donde:

- (S_1) y (S_2) son las representaciones vectoriales de las dos oraciones.
- (W) y (b) son los parámetros aprendidos por la red.
- (y) es la probabilidad de que (S_2) sea la oración siguiente de (S_1)

2.2.4.3 Estructura BERT

BERT emplea la arquitectura Transformer basada en múltiples capas de autoatención, lo que le permite capturar relaciones contextuales en el texto de manera más efectiva. Cada token de entrada es procesado mediante:

- **Embeddings:** Se representan las palabras como vectores de dimensión fija, combinando información de tokens, posiciones y segmentos.
- **Capas de Transformadores:** Se aplican mecanismos de autoatención que permiten que cada palabra se relacione con cualquier otra en la oración.
- **Capa de salida:** Dependiendo de la tarea (clasificación, generación, etc.), se usa una capa diferente.

La ecuación clave para el mecanismo de self-attention en BERT es:

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

Donde:

- (Q, K, V) son las matrices de consulta, clave y valor, respectivamente.
- (d_k) es la dimensión de las claves.
- La operación (QK^T) calcula la similitud entre cada token en la secuencia.

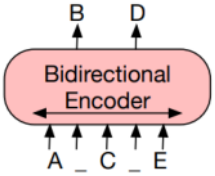
- La función softmax normaliza los valores para obtener pesos de atención.

2.2.5 BART (Bidirectional and Auto-Regressive Transformers)

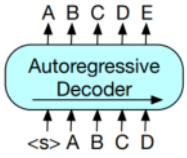
BART (Bidirectional and Auto-Regressive Transformers) es un modelo de lenguaje basado en la arquitectura Transformer que combina características de modelos bidireccionales como BERT y modelos autoregresivos como GPT (Lewis et al., 2019), como se muestra en la figura 3. Su principal fortaleza radica en su capacidad de generar texto mediante una arquitectura codificador-decodificador *encoder-decoder*, como se puede apreciar en la Figura 3, lo que lo hace particularmente útil para tareas de generación de texto, como el resumen automático y la traducción.

Figura 3

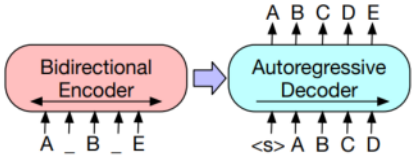
Comparación entre BART con BERT y GPT



(a) BERT: Se reemplazan tokens aleatorios con máscaras y el documento se codifica bidireccionalmente.



(b) GPT: Predice tokens de forma autoregresiva, lo que le permite generar texto.



(c) BART: Permite transformaciones arbitrarias de ruido en las entradas del codificador, asegurando una mejor reconstrucción de texto.

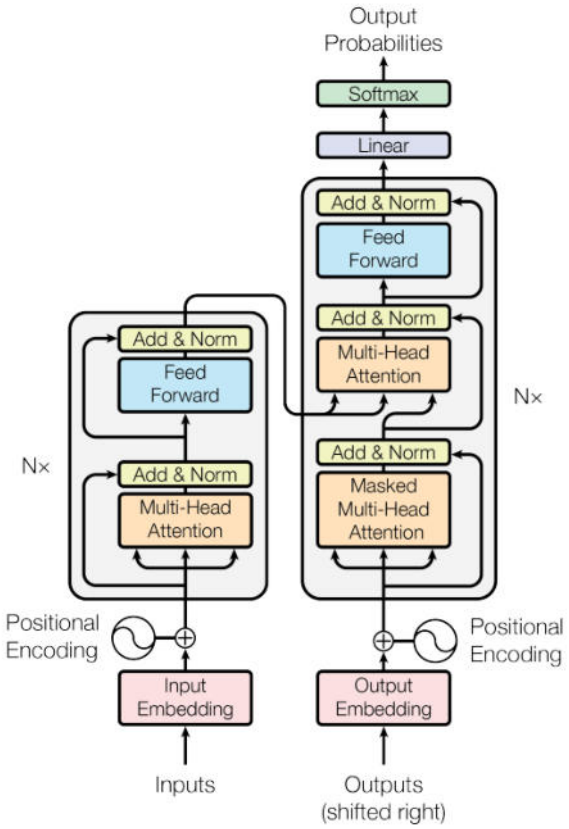
Nota. BERT utiliza un enfoque bidireccional para el aprendizaje contextual, prediciendo tokens enmascarados de manera independiente. BART combina ambas estrategias con un codificador bidireccional y un decodificador autoregresivo, lo que le permite reconstruir textos corrompidos y mejorar la generación de texto (Lewis et al., 2019).

2.2.5.1 Arquitectura BART

La arquitectura consiste en un codificador (Encoder), se encarga de procesar la secuencia de entrada y aprender una representación contextual bidireccional de las palabras. Decodificador (Decoder), genera la secuencia de salida de manera autoregresiva, prediciendo cada token secuencialmente basándose en los tokens previos.

Figura 4

Arquitectura Transformers



Nota. Consiste en un codificador y un decodificador, ambos basados en múltiples capas de autoatención y redes feed-forward. El mecanismo de autoatención permite a cada token ponderar la importancia de los demás en la secuencia, facilitando la captura de dependencias a largo plazo. Esta estructura optimiza tareas de procesamiento de lenguaje natural como traducción automática, generación de texto y resumen automático (Vaswani et al., 2023).

- **Funcionamiento del Codificador:** El codificador de BART funciona de manera similar a BERT, permitiendo que la representación de cada palabra dependa de todas las demás en la oración. Durante el preentrenamiento, el codificador recibe entradas que han sido corrompidas

intencionalmente mediante diferentes estrategias de ruido (denoising), lo que entrena al modelo a reconstruir el texto original.

- **Funcionamiento del Decodificador:** El decodificador de BART opera como un modelo autoregresivo, similar a GPT. Predice cada palabra de salida de manera secuencial, utilizando como contexto tanto la información proporcionada por el codificador como los tokens generados previamente.

  Matemáticamente, la generación de tokens en el decodificador se modela como:

$$P(y_t \mid y_{1:t-1}, x) = \text{softmax}(W h_t)$$

  Donde:

  - (y_t) es el token generado en el tiempo (t).
  - $(y_{1:t-1})$ representa los tokens previos generados por el decodificador.
  - (x) es la secuencia de entrada original procesada por el codificador.
  - (h_t) es la representación oculta del decodificador en el tiempo (t).
  - (W) es una matriz de pesos aprendidos que proyecta (h_t) al espacio de probabilidad de tokens de salida.

2.2.5.2 Preentrenamiento de BART

El preentrenamiento de BART se basa en un enfoque de denoising, en el que la entrada se corrompe mediante diversas técnicas y el modelo aprende a reconstruir la versión original. Entre las principales técnicas de corrupción utilizadas se encuentran:

- **Eliminación de tokens:** Se eliminan palabras al azar del texto.
- **Permutación de palabras:** Se reordena aleatoriamente el orden de las palabras.
- **Sustitución de spans de texto con [MASK]:** Similar al enmascaramiento de BERT, pero aplicado a fragmentos enteros de texto.

- **Eliminación de oraciones enteras:** Se eliminan frases completas, forzando al modelo a predecir información faltante a nivel global.

La función de pérdida durante el preentrenamiento se basa en minimizar la probabilidad negativa logarítmica de la reconstrucción del texto original:

$$L = -\sum_{i=1}^{N} \log P\left(x_i \mid \tilde{x}\right)$$

Donde:

- (x_i) representa la palabra original en la posición (i).
- $(\tilde{x})$ es la versión corrompida de la secuencia de entrada.
- $(P(x_i \mid \tilde{x}))$ es la probabilidad de que el modelo prediga correctamente la palabra (x_i).

2.2.5.3 Ajuste Fino (Fine Tuning)

Después del preentrenamiento, BART se ajusta (fine-tuning) en tareas específicas con datos etiquetados. Para el ajuste fino en generación de resúmenes, el modelo se entrena con pares de texto donde la entrada es un documento completo y la salida es su resumen correspondiente

En esta fase, la función de pérdida se expresa como:

$$L = -\sum_{t=1}^{T} \log P\left(y_t \mid y_{1:t-1}, x\right)$$

Donde:

- (y_t) es el token generado en el instante (t).
- $(y_{1:t-1})$son los tokens previos generados.
- (x) es el texto original de entrada.

2.2.5.4 Mecanismos de Atención

BART emplea el mecanismo de self-attention tanto en el codificador como en el decodificador, permitiendo que cada token de entrada influya en la representación de los demás. Además, incorpora atención cruzada en el decodificador, lo que permite que la generación de tokens se guíe por la información proveniente del codificador.

El mecanismo de atención cruzada se define como:

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

Donde:

- $(\mathrm{Q}, \mathrm{K}, \mathrm{V})$ son las matrices de consulta, clave y valor, respectivamente.
- $(\mathrm{d_k})$ es la dimensión de las claves.
- La operación $(\mathrm{QK^T})$ calcula la similitud entre cada token de entrada y cada token de salida generado.
- La función softmax normaliza los valores para obtener pesos de atención.

2.2.6 Técnicas de Resumen Automático

El resumen automático busca condensar información de uno o varios documentos en un texto más corto que conserve los puntos clave. Se clasifica en:

- **Resumen extractivo:** Este enfoque identifica y extrae las oraciones más importantes del texto original, combinándolas para formar el resumen. Es como seleccionar las piezas de un rompecabezas que mejor representan la imagen completa. La ventaja principal de este método es su sencillez y la garantía de que el resumen resultante sea gramaticalmente correcto, ya que utiliza oraciones existentes. Sin embargo, puede carecer de cohesión y fluidez si las oraciones extraídas no se relacionan entre sí de manera natural.

- **Resumen abstractivo:** Este enfoque, más ambicioso, busca generar un resumen que capture la esencia del texto original, pero que puede contener palabras y frases que no aparecen en el documento fuente. Es como reescribir la historia con tus propias palabras, manteniendo la fidelidad a los hechos. La abstracción permite una mayor flexibilidad y la posibilidad de generar resúmenes más concisos y coherentes. No obstante, presenta desafíos significativos, como la necesidad de comprender el significado profundo del texto y la posibilidad de generar resúmenes gramaticalmente incorrectos o que distorsionen la información original.

2.2.6.1 Métodos tradicionales y estadísticos

TF-IDF (Term Frequency-Inverse Document Frequency): Pondera la importancia de palabras en un documento en relación con un corpus. Las palabras que son frecuentes en un documento pero raras en otros se consideran más relevantes.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Donde:

- $\text{F}(\text{t}, \text{d})$ es la frecuencia del término t en el documentog.
- $\text{IDF}(\text{t})$ es la frecuencia inversa del documento para el término t, que se calcula como:

$$\text{IDF}(\text{t}) = \log \quad \frac{\text{N}}{\text{DF}(\text{t})}$$

Donde N es el número total de documentos en el corpus DF(t) es el número de documentos que contienen el término t.

2.2.6.2 Text Bank

Construye un grafo de oraciones donde las conexiones representan similitud semántica. Las oraciones con mayor importancia en el grafo se seleccionan para el resumen. La importancia de un nodo (oración) se calcula iterativamente usando una fórmula similar a PageRank:

$$\text{Score}(V_i) = (1 - d) + d \sum_{V_j \in \text{Adj}(V_i)} \frac{\text{Score}(V_j)}{\deg(V_j)}$$

Donde:

- $\text{Score}(V_i)$ es la puntuación del nodo V_i (oración).
- d es un factor de amortiguación (típicamente 0.85).
- $\text{Adj}(V_i)$es el conjunto de nodos adyacentes a V_i.
- $\deg(V_j)$ es el grado del nodo V_i.

Estos métodos se basan en características superficiales del texto y pueden tener dificultades para capturar relaciones semánticas profundas.

2.2.7 Evaluación de Resúmenes automáticos

La evaluación de resúmenes automáticos es un componente crítico para determinar la calidad de los sistemas de generación de resúmenes. En el ámbito del Procesamiento de Lenguaje Natural, se han propuesto diversas métricas automáticas que cuantifican la similitud entre un resumen generado y uno o varios resúmenes de referencia. Estas métricas se basan en conceptos matemáticos que permiten medir aspectos como la superposición de n-gramas, la similitud semántica y la cohesión global del texto.

2.2.7.1 Métricas de evaluación

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** es un conjunto de métricas que miden la superposición de n-gramas entre el resumen generado y el resumen de referencia. En particular, ROUGE-N se define como:
    - $\text{ROUGE-N} = \frac{\sum_{S \in \{\text{Resúmenes de Referencia}\}} \sum_{\text{gram}_n \in S} \text{Count}_{match}(\text{gram}_n)}{\sum_{S \in \{\text{Resúmenes de Referencia}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}$
    - Donde:
        - $(\text{Count}_{\text{match}}(\text{gram}_\text{n}))$ es el número de n-gramas que coinciden entre el resumen generado y el de referencia.
        - $(\text{Count}(\text{gram}_\text{n}))$ es el total de n-gramas en el resumen de referencia.
- **BLEU:** Originalmente diseñado para la evaluación de traducciones automáticas, BLEU (Bilingual Evaluation Understudy) compara la precisión de n-gramas del resumen generado respecto al de referencia. Su fórmula se expresa de la siguiente manera:
    - $\text{BLEU} = \text{BP} \cdot \exp(\sum_{n=1}^{N} w_n \log p_n)$
    - Donde:
        - (p_n) representa la precisión de n-gramas.
        - (w_n) son los pesos asignados a cada orden de n-grama (usualmente iguales).
        - (BP) es la penalización por brevedad (Brevity Penalty), que evita que se obtengan puntuaciones altas para resúmenes demasiado cortos.

2.2.8 Características y desafíos del Lenguaje legal

El lenguaje legal se distingue por su alta complejidad y formalidad, aspectos que lo diferencian notablemente del lenguaje cotidiano. En este ámbito, la precisión terminológica es esencial para asegurar que cada norma, precedente o doctrina se interprete de forma correcta y sin ambigüedades. Esta necesidad se traduce en oraciones extensas, estructuras jerárquicas

complejas y el uso de una terminología altamente especializada, lo que hace que la representación computacional de dichos textos sea especialmente desafiante.

Por un lado, la complejidad del lenguaje legal implica que los textos deben capturar con exactitud relaciones lógicas y argumentativas, garantizando la coherencia y el rigor en la comunicación jurídica. Estas características requieren que cualquier sistema automatizado de procesamiento y resumen de textos legales sea capaz de manejar no solo el análisis sintáctico y semántico, sino también la estructura discursiva y la interrelación de conceptos clave.

Por otro lado, la diversidad en la redacción de resoluciones judiciales, especialmente en contextos locales como el sistema judicial peruano, presenta un reto adicional. La heterogeneidad en estilos, vocabulario y formatos dificulta la aplicación directa de modelos entrenados con datos de otras jurisdicciones, lo que subraya la necesidad de disponer de conjuntos de datos etiquetados específicos para el dominio legal local (Hennis, 2023). Sin estos recursos, los modelos podrían no capturar las sutilezas lingüísticas y normativas propias de cada jurisdicción, afectando la calidad y precisión de los resúmenes generados.

Además, las implicaciones éticas y de calidad asociadas con la generación automática de resúmenes en el ámbito legal son de gran relevancia. La posibilidad de que un sistema automatizado omita matices críticos o introduzca errores interpretativos puede tener consecuencias significativas en la toma de decisiones y en la transparencia del proceso judicial. La dependencia excesiva en estas tecnologías, sin la adecuada supervisión y validación por parte de expertos humanos, podría erosionar la confianza en los sistemas automatizados y, en última instancia, afectar la integridad de la administración de justicia (cited in Wired), 2024).

En síntesis, la comprensión profunda de las características del lenguaje legal, junto con la identificación de los desafíos asociados a su adaptación y evaluación, es fundamental para el

desarrollo de sistemas robustos de generación automática de resúmenes que sean capaces de operar de manera eficaz y ética en contextos jurídicos complejos.

2.3 Definición de términos

- **Atención Multi-Cabeza:** Técnica utilizada en modelos Transformer para procesar simultáneamente diferentes representaciones del texto, facilitando la identificación de información clave en documentos extensos (Vaswani et al., 2017).
- **Corpus Jurídico:** Conjunto de documentos legales utilizados como base de datos para entrenar modelos de PLN en el ámbito judicial. Puede incluir resoluciones judiciales, leyes, reglamentos y doctrinas (Ashley, 2017).
- **Explicabilidad en PLN:** Capacidad de un modelo de PLN para justificar sus predicciones o decisiones, esencial en el ámbito jurídico donde la transparencia y la interpretabilidad son fundamentales (Doshi-Velez & Kim, 2017).
- **Fine-Tuning:** Proceso de ajuste de un modelo preentrenado en un dominio específico, en este caso, el ámbito legal, para mejorar su rendimiento en tareas concretas como la generación de resúmenes de resoluciones judiciales (Howard & Ruder, 2018).
- **Modelo de Lenguaje Preentrenado:** Algoritmo basado en redes neuronales profundas que ha sido entrenado en grandes volúmenes de texto para comprender y generar lenguaje de manera contextualizada (Devlin et al., 2018), como BERT o GPT.
- **Memoria a Largo Plazo en PLN:** Capacidad de ciertos modelos, como LSTM o Transformers, de retener información relevante durante largas secuencias de texto (Hochreiter & Schmidhuber, 1997), permitiendo captar relaciones entre conceptos distantes en un documento legal.

- **Precedente Judicial:** Decisión emitida por una instancia superior que sirve como referencia obligatoria o persuasiva para la resolución de casos similares en el futuro (Holmes Jr, 1881).
- **Resolución Judicial:** Documento emitido por un juez o tribunal en el que se fundamenta una decisión sobre un caso específico. Contiene argumentaciones legales, precedentes y conclusiones que impactan en la jurisprudencia quien enfatiza la importancia del razonamiento jurídico en las resoluciones (Nino, 1996).
- **Sesgo en Modelos de IA para Derecho:** Tendencia de los modelos a generar resultados influenciados por datos desbalanceados o incompletos, lo que puede afectar la imparcialidad de los resúmenes legales generados automáticamente (Barocas et al., 2019).
- **Tokenización Jurídica:** Proceso de segmentación de un texto legal en unidades mínimas significativas (tokens), adaptado para reconocer estructuras propias del lenguaje legal, como referencias a normativas o artículos de ley (Zheng et al., 2020).

III. Metodología

3.1 Tipo y nivel de investigación

3.1.1 Tipo de investigación

La presente investigación se situó en el tipo de investigación aplicada, ya que su propósito fue desarrollar y validar los modelos de resúmenes basado en técnicas de procesamiento de lenguaje natural para generar resúmenes automáticos de resoluciones judiciales con el objetivo de optimizar el acceso y análisis de estos documentos en el ámbito judicial. Asimismo, se evaluó el desempeño de los modelos mediante métricas objetivas como ROUGE y BLEU; estas métricas permitieron evaluar la calidad de los resúmenes textuales de los enfoques abstractivos aplicados a las resoluciones judiciales.

3.1.2 Nivel de investigación

El estudio se situó en un nivel de investigación explicativo-comparativo, puesto que buscó analizar y comprender cómo distintos modelos de generación automática de resúmenes incidieron en la síntesis de información legal. Asimismo, se examinaron los factores que influyeron en la calidad de los productos generados y se evaluó qué enfoque abstractivo ofreció mejores resultados dentro del contexto jurídico laboral.

3.2 Ámbito temporal y espacial

La delimitación del estudio constituye un componente metodológico esencial, define las fronteras del corpus documental, contextualiza el entorno de entrenamiento de los modelos y establece el marco interpretativo de los resultados. La presente investigación se ha circunscrito a los siguientes criterios:

3.2.1 Ámbito Espacial

El estudio se situó geográficamente en la ciudad de Lima, restringiendo la fuente de datos a una única entidad judicial: la Corte Superior de Justicia de Lima. La totalidad del corpus documental procesado proviene de los repositorios digitales de esta jurisdicción específica. Esta decisión metodológica garantiza la homogeneidad en el formato de las resoluciones y en los criterios de redacción propios de dicha corte, reduciendo el ruido que podría introducir la mezcla de estilos de diferentes distritos judiciales.

3.2.2 Ámbito Temático

La investigación estuvo enfocado de manera exclusiva en la Especialidad Laboral. Esta delimitación temática es crítica para la especialización de los modelos de inteligencia artificial (léxicos, semánticos e híbridos), los cuales fueron entrenados y evaluados asimilando la terminología técnica, los fundamentos de derecho y las estructuras argumentativas inherentes a las resoluciones de apelación en este campo del derecho.

3.2.3 Ámbito Temporal

El horizonte temporal de la investigación abarcó un período de dos años. Todas las resoluciones de apelación que conformaron el corpus documental fueron emitidas y publicadas entre el año 2021 y el año 2022. Esta acotación asegura que los modelos sean entrenados con jurisprudencia reciente, reflejando el estado actual de la doctrina procesal laboral.

3.3 Población y muestra

La investigación se desarrolló durante el periodo marzo a noviembre de 2025, abarcando las fases de: recopilación de datos, desarrollo del modelo, entrenamiento, evaluación y validación. Las resoluciones judiciales consideradas corresponderán al periodo 2021 al 2022, permitiendo

asegurar una base de datos reciente, variada y representativa de las decisiones judiciales emitidas por la Corte Superior de Lima, Perú.

3.3.1 Población

La población de esta investigación estuvo conformada por el conjunto de resoluciones judiciales emitidas por la Corte Superior de Lima, Perú en diversas materias legales. Estas resoluciones contienen información detallada sobre casos judiciales, fundamentos legales y decisiones tomadas por los jueces. Debido a la gran cantidad de documentos generados anualmente, el análisis de la totalidad de la población resulta inviable, por lo que se seleccionará una muestra representativa.

3.3.2 Muestra

La muestra definitiva para la presente investigación estuvo constituida por un corpus especializado de 6,848 resoluciones judiciales emitidas por las Cortes Superiores de Justicia del Perú. La selección de estos documentos se realizó priorizando la disponibilidad de acceso a formatos digitales procesables y la representatividad de la terminología jurídica nacional.

El análisis estadístico descriptivo del corpus permitió revelar la alta variabilidad y complejidad de los documentos procesados. La longitud promedio de las resoluciones originales es de 5,955 palabras con un rango que oscila desde documentos breves de 28 palabras hasta sentencias extensas de 38,472 palabras. Por otro lado, los resúmenes de referencia presentan una longitud media de 484 palabras, lo que implica una tasa de compresión significativa que el modelo debe aprender a replicar.

Para garantizar la calidad de los datos, la muestra fue depurada aplicando criterios de inclusión y exclusión, eliminando registros corruptos o vacíos. Posteriormente, el corpus total de 6,848

documentos fue particionado aleatoriamente siguiendo la proporción estándar 80/10/10, garantizando conjuntos disjuntos para asegurar la imparcialidad de la evaluación.

Tabla 2

Distribución de la muestra por subconjuntos de datos

Subconjunto	Cantidad de Documentos	Porcentaje	Propósito
Entrenamiento (Train)	5,478	80%	Ajuste de los pesos sinápticos de los modelos (*Fine-tuning*).
Validación (Val)	685	10%	Monitoreo de la convergencia y ajuste de hiperparámetros.
Prueba (Test)	685	10%	Evaluación final ciega para obtener las métricas ROUGE/BLEU.
TOTAL	**6,848**	**100%**	

Nota. Elaboración propia.

3.4 Instrumentos

Para la ejecución experimental, el entrenamiento de los modelos y la evaluación de resultados, se emplearon instrumentos de hardware y software especializados en computación de alto rendimiento y procesamiento de lenguaje natural. A continuación, se detallan las especificaciones técnicas:

1) **Recursos Computacionales (Hardware):** El entrenamiento de los modelos de aprendizaje profundo (Deep Learning) se llevó a cabo utilizando la plataforma en la nube Google Colab Pro, aprovechando su infraestructura de aceleración por GPU. Las especificaciones promedio del entorno de ejecución fueron:
    a) Procesador (GPU): NVIDIA Tesla T4 (o A100 según disponibilidad) con 32 GB de VRAM, esencial para soportar la carga de tensores y el cálculo de gradientes en modelos Transformer.

b) Memoria RAM: 12 GB a 25 GB de memoria del sistema, destinada a la manipulación de los DataFrames y preprocesamiento del corpus.

c) Almacenamiento: Google Drive integrado para la persistencia de los checkpoints de los modelos y el almacenamiento del corpus judicial.

2) **Herramientas de Software y Librerías:** El desarrollo se realizó íntegramente en el lenguaje de programación Python (v3.10+), utilizando las siguientes librerías fundamentales del ecosistema de Ciencia de Datos:

a) PyTorch: Framework base para la construcción y ejecución de las redes neuronales, manejo de tensores y cálculo de la función de pérdida.

b) Hugging Face Transformers: Biblioteca central utilizada para descargar, configurar y realizar el ajuste fino (fine-tuning) de las arquitecturas pre-entrenadas (BART, mT5, Pegasus, BETO).

c) Pandas y NumPy: Instrumentos para la manipulación estructurada del dataset, limpieza de datos y operaciones vectoriales.

d) PyPDF2: Herramienta utilizada para la extracción y digitalización del texto contenido en las resoluciones judiciales en formato PDF.

e) Scikit-learn: Empleada para la partición estratificada del corpus en conjuntos de entrenamiento, validación y prueba.

3) **Modelos Pre-entrenados (Artefactos):** Como instrumentos base para la transferencia de aprendizaje, se utilizaron los siguientes puntos de control (checkpoints) públicos:

a) facebook/bart-large-cnn: Modelo base para la arquitectura BART.

b) google/mt5-small: Modelo base multilingüe para la arquitectura mT5.

c) google/pegasus-xsum: Modelo base para la arquitectura Pegasus.

d) dccuchile/bert-base-spanish-wwm-cased (BETO): Modelo base para la adaptación de la arquitectura BERT al español.

4) **Instrumentos de Medición (Métricas):** Para la cuantificación objetiva de la calidad de los resúmenes, se implementaron las librerías oficiales de evaluación:
   a) ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Instrumento para medir la cobertura y recuperación de n-gramas (ROUGE-1, ROUGE-2, ROUGE-L).
   b) BLEU (Bilingual Evaluation Understudy): Instrumento para medir la precisión y fluidez de la generación en comparación con los resúmenes de referencia.

3.5 Procedimientos

El estudio se desarrolló con un diseño de tipo experimental, ya que se implementaron y evaluaron diferentes modelos de generación automática de resúmenes de resoluciones judiciales. Se efectuaron pruebas controladas para medir su rendimiento utilizando métricas cuantitativas (ROUGE y BLEU), lo que permitió comparar sistemáticamente el enfoque abstractivo.

Dentro del enfoque experimental, la investigación permaneció con un diseño cuasi-experimental con grupos no equivalentes y medición pretest-postest, dado que se trabajó con un corpus real de resoluciones emitidas por la Corte Superior de Lima, Perú. No se manipularon las variables en un entorno totalmente controlado, en su lugar se observó el desempeño de los modelos en condiciones cercanas a un entorno real, sin intervenir en la redacción original de los documentos judiciales.

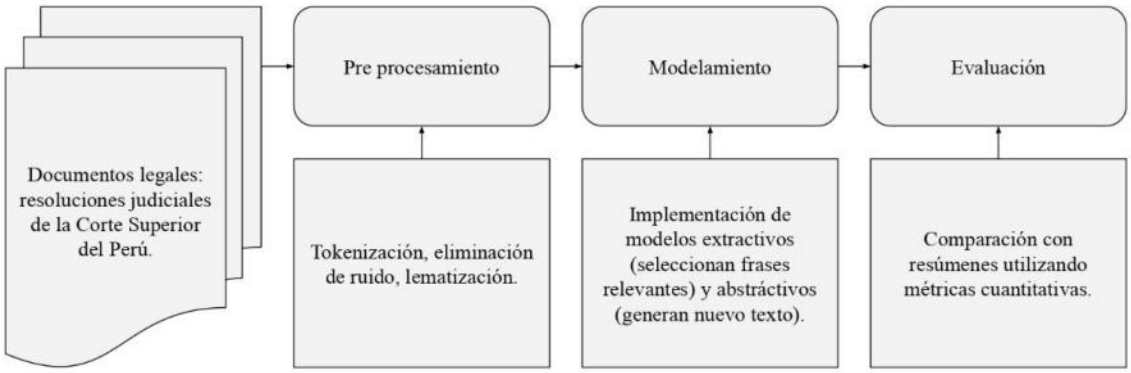
- **Grupo Experimental.-** Estuvo conformado por resoluciones judiciales procesadas con los modelos de inteligencia artificial desarrollados en este estudio. A este grupo se le aplicó la intervención experimental, es decir, la generación automática de resúmenes mediante IA.

- **Grupo Control.-** Estuvo compuesto por resoluciones judiciales con resúmenes elaborados por sistemas básicos (baseline) ya existentes. Este grupo se utilizó como referencia para comparar el desempeño de los modelos propuestos.

El proceso experimental contempló las fases representadas en la figura correspondiente, basado en las investigaciones tituladas: Effective Deep Learning Approaches for Summarization of Legal Texts (Anand & Wagh, 2022) y Evaluation of Automatic Legal Text Summarization Techniques for Greek Case Law (Koniaris et al., 2023).

Figura 5

Metodología



Nota. Elaboración propia.

Este diseño permitió obtener conclusiones fundamentadas sobre la viabilidad y precisión de los enfoques utilizados para la generación de resúmenes automáticos en el ámbito judicial.

3.5.1 Construcción del Corpus

A continuación, se sistematiza el desarrollo de todas las etapas experimentales, siguiendo la estructura metodológica definida previamente en el proyecto. Esta sección tiene como objetivo transparentar el proceso técnico, describiendo las decisiones de ingeniería, la configuración de los entornos de hardware y software, y los procedimientos aplicados para la generación y evaluación de los resúmenes judiciales.

3.5.1.1 Recuperación de documentos legales

La consolidación de un corpus estructurado y de alta calidad constituye el cimiento indispensable para el desarrollo exitoso de cualquier sistema de Procesamiento de Lenguaje Natural (PLN). En el dominio jurídico, esta fase adquiere una relevancia crítica dada la complejidad sintáctica y la densidad terminológica propia de los textos legales. Por consiguiente, el objetivo de este procedimiento fue conformar un conjunto de datos representativo, depurado e íntegro, que garantizara una base sólida para el entrenamiento y la evaluación objetiva de los modelos de resumen automático propuestos.

Para la adquisición de los documentos, se recurrió como fuente primaria al portal oficial de Jurisprudencia Nacional Sistematizada del Poder Judicial del Perú (https://jurisprudencia.pj.gob.pe/jurisprudenciaweb/). Este repositorio digital centraliza las resoluciones emitidas por las diversas cortes del país, constituyendo la fuente de verdad más fiable y actualizada para los fines de la presente investigación.

Con el propósito de asegurar la viabilidad del estudio y mantener la coherencia semántica del corpus, se adoptó la decisión metodológica de circunscribir el alcance de la muestra a la Especialidad Laboral. La justificación para esta segmentación temática se fundamenta en tres criterios técnicos principales:

1. **Volumen de Datos y Densidad Muestral:**

   El Derecho Laboral representa una de las áreas con mayor carga procesal en el sistema judicial peruano, lo que garantiza un flujo constante y masivo de resoluciones. Esta abundancia de datos textuales es un requisito indispensable para el entrenamiento eficaz de arquitecturas basadas en Transformers, las cuales son "intensivas en datos" y requieren

corpus extensos para calibrar sus millones de parámetros, minimizando así el riesgo de sobreajuste (overfitting) durante la fase de aprendizaje.

2. **Estructura y Patrones Recurrentes**

   Si bien la casuística es diversa, los litigios laborales tienden a orbitar en torno a núcleos conceptuales definidos (despidos arbitrarios, pago de beneficios sociales, reposiciones). Esta recurrencia temática presenta un escenario idóneo para que los modelos generativos aprendan a identificar patrones argumentativos estables. Al existir una estructura lógica repetitiva en los fundamentos y fallos, el modelo puede abstraer con mayor facilidad las secciones críticas que deben ser preservadas en el resumen, diferenciándolas de la argumentación accesoria.

3. **Control del Dominio Lingüístico**

   La restricción del alcance a una única especialidad permitió trabajar con un espacio vectorial semánticamente homogéneo. Al evitar la contaminación con terminología propia de ramas dispares (como el Derecho Penal o Civil), se reduce significativamente el *ruido semántico*. Esto facilitó que el decodificador del modelo prediga secuencias de texto con mayor precisión y coherencia, utilizando la jerga jurídica laboral correcta sin caer en ambigüedades polisémicas.

Figura 6

Interfaz de la plataforma de Jurisprudencia Nacional Sistematizada, fuente primaria de recolección



Nota. Captura del portal oficial del Poder Judicial del Perú.

Finalmente, para garantizar la representatividad estadística de la muestra, se procedió a la descarga y procesamiento de documentos abarcando todas las sub-materias disponibles dentro de la especialidad Laboral. Esta estrategia de cobertura total aseguró que el corpus final reflejara fielmente la diversidad real de la carga procesal, evitando sesgos temáticos hacia un tipo específico de controversia.

3.5.1.2 Metodología de extracción automatizada en dos fases

Dada la arquitectura dinámica de la plataforma web del Poder Judicial, se diseñó e implementó un pipeline de extracción de datos (web scraping) secuencial, dividido en dos fases estratégicas. Este enfoque desacoplado permitió asegurar la integridad referencial de los datos y minimizar fallos de conexión durante la descarga masiva.

Fase 1: Rastreo y Extracción de Metadatos Estructurados

En esta etapa inicial, se desarrolló un algoritmo de navegación sistemática utilizando las librerías de Python Scrapy y BeautifulSoup. El objetivo fue indexar la totalidad de los expedientes disponibles sin descargar aún los documentos pesados. El proceso automatizado siguió los siguientes pasos:

a) **Filtrado de Dominio:** El script aplicó una restricción de búsqueda en la interfaz del portal, limitando los resultados exclusivamente a la etiqueta "Especialidad: Laboral".
b) **Iteración y Recolección de Atributos:** El algoritmo recorrió paginadamente los resultados, extrayendo de cada entrada los metadatos visibles. Se definieron como atributos clave:
    i) Apelación (ID): Utilizado como clave primaria para la identificación única del caso.
    ii) Especialidad y Materia: Etiquetas necesarias para el análisis de distribución del corpus.
c) **Captura de Enlace Fuente:** Simultáneamente a los metadatos, se extrajo y almacenó la URL directa al documento fuente (predominantemente en formato PDF), creando un puente lógico entre el registro y su contenido.

El producto de esta fase fue un archivo CSV preliminar que funcionó como un índice maestro, conteniendo las coordenadas de acceso y los metadatos de todos los casos identificados.

Fase 2: Ingesta Documental y Procesamiento de Texto

Con el índice maestro consolidado, se ejecutó la segunda fase del pipeline, orientada a la adquisición del contenido no estructurado (texto de las sentencias):

a) **Lectura del Índice**: Un segundo módulo de software cargó el archivo CSV generado en la fase anterior para iterar sobre la lista de URLs válidas.

b) **Descarga Programática:** Para cada registro, el sistema descargó el archivo PDF correspondiente, almacenándolo temporalmente y vinculándolo a su ID de expediente (Apelación).
c) **Parsing y Extracción de Texto (OCR/Texto Plano):** Utilizando la librería PyPDF2, se procesó cada binario PDF para extraer su capa de texto. El contenido crudo fue capturado en memoria, descartando imágenes o elementos no textuales.

La etapa final consistió en la fusión lógica de los resultados de ambas fases. El texto íntegro extraído de los PDFs se alineó con su respectiva fila de metadatos utilizando el ID de Apelación como llave foránea. Este procedimiento culminó en la creación del corpus unificado (corpus_final_resumido.csv), una estructura tabular donde cada fila representa una resolución judicial completa (Metadatos + Texto Completo). Este dataset enriquecido constituyó la materia prima para las subsiguientes etapas de preprocesamiento, limpieza y entrenamiento de los modelos generativos.

3.5.1.3 Exploración de los datos

Tras la consolidación del corpus documental, se ejecutó una fase rigurosa de Análisis Exploratorio de Datos (EDA) con el propósito de caracterizar las propiedades estadísticas de la muestra, auditar su calidad y acondicionar los datos para las etapas subsiguientes de entrenamiento y evaluación de los modelos generativos.

Previo al proceso de modelado, se realizó una auditoría de integridad sobre el conjunto de datos final, constituido por 6,848 registros de resoluciones judiciales. El hallazgo más relevante de esta inspección preliminar fue la completitud absoluta de la información. Se verificó matemáticamente la inexistencia de valores nulos (missing values) en la totalidad de las dimensiones que componen el dataset.

Esta integridad estructural representa una ventaja metodológica sustancial para la investigación, pues elimina la necesidad de aplicar técnicas sintéticas de imputación de datos o la reducción de la muestra por eliminación de filas incompletas (drop-na). En consecuencia, se garantiza que los modelos de Inteligencia Artificial se nutrirán del 100% de la información recopilada, sin sesgos inducidos por pérdida de datos.

Figura 7

Vista preliminar del dataframe con el corpus de jurisprudencias procesado.

[7]:

	Apelación	Especialidad	Materia	texto_completo	Texto_resumido
0	Apelación 000011-2021	Especialidad: Laboral	Materia: Contratos	CORTE SUPERIOR DE JUSTICIA DE LIMA\nCUARTA SAL...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**CUAR...
1	Apelación 000016-2022	Especialidad: Laboral	Materia: Reposición	8° SALA LABORAL\nEXPEDIENTE : 00016-2022-0-180...	Este documento corresponde al encabezado (cará...
2	Apelación 000030-2020	Especialidad: Laboral	Materia: Indemnización Laboral	CORTE SUPERIOR DE JUSTICIA DE LIMA\nPRIMERA SA...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**PRIM...
3	Apelación 000031-2019	Especialidad: Laboral	Materia: Pago de Beneficios Sociales	CORTE SUPERIOR DE JUSTICIA DE LIMA\nPRIMERA SA...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**PRIM...
4	Apelación 000037-2018	Especialidad: Laboral	Materia: Pago de Beneficios Sociales	CORTE SUPERIOR DE JUSTICIA DE LIMA\nPRIMERA SA...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**PRIM...
...	...	...	...	...	...
6843	Queja 000223-2019	Especialidad: Laboral	Materia: Incumplimiento de Disposiciones y Nor...	1° SALA LABORAL PERMANENTE\nEXPEDIENTE : 00223...	**RESUMEN DE RESOLUCIÓN JUDICIAL**\n\n**EXPEDI...
6844	Queja 000233-2019	Especialidad: Laboral	Materia: Pago de Beneficios Sociales	CORTE SUPERIOR DE JUSTICIA DE LIMA\nPRIMERA SA...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**PRIM...
6845	Queja 000376-2019	Especialidad: Laboral	Materia: Ofrecimiento de Pago	1° SALA LABORAL PERMANENTE\nEXPEDIENTE : 00376...	**EXPEDIENTE:** 00376-2019-0-1801-SP-LA-01\n**...
6846	Queja 000381-2019	Especialidad: Laboral	Materia: Contratos	CORTE SUPERIOR DE JUSTICIA DE LIMA\nPRIMERA SA...	**CORTE SUPERIOR DE JUSTICIA DE LIMA**\n**PRIM...
6847	Queja 000408-2019	Especialidad: Laboral	Materia: Incumplimiento de Disposiciones y Nor...	CORTE SUPERIOR DE JUSTICIA DE LIMA\nSÉPTIMA SA...	**EXPEDIENTE:** N° 00408-2019-0-1801-JR-LA-60 ...

6848 rows × 5 columns

[8]:
```
df.to_csv('corpus_final_resumido.csv', index=False)
```

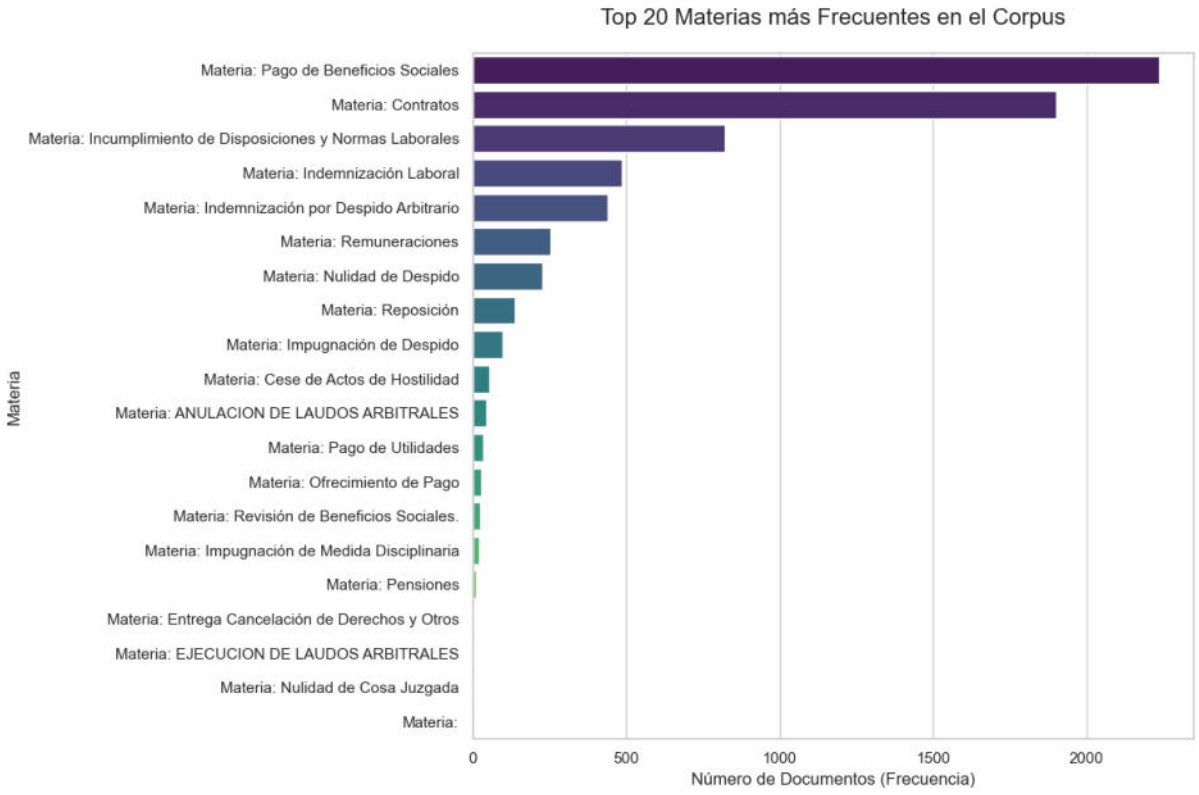
Nota. Elaboración propia.

Adicionalmente, se validó la consistencia del esquema de datos (schema validation). Se confirmó que las columnas que albergan información semántica crítica específicamente texto_completo (entrada), texto_resumido (objetivo) y materia (contexto) fueron interpretadas correctamente por el entorno de análisis como tipos de dato objeto (string/object). Esta correcta tipificación asegura que las operaciones de tokenización y vectorización posteriores se ejecuten sin errores de formato. Dado que la inferencia de tipos fue exitosa, no se requirieron procesos de casting o conversión forzada en esta etapa.

Habiendo certificado la calidad técnica, la integridad y la consistencia estructural del corpus, se da por concluida la fase de verificación de datos. El análisis exploratorio procedío a examinar el contenido semántico y la distribución temática de los documentos. Este estudio comenzó con la variable Materia, identificada como el metadato rector para comprender la diversidad casuística que los modelos deberán aprender a sintetizar, los resultados de puede apreciar en la Figura 8.

Figura *8*

Distribución de frecuencias de las materias legales en el corpus



Nota. Elaboración propia.

3.5.2 Pre procesamiento

El pre-procesamiento fue una fase crítica en el flujo de trabajo de Procesamiento de Lenguaje Natural (NLP), actuando como el puente necesario entre la naturaleza no estructurada y ruidosa de las resoluciones judiciales originales y los estrictos requerimientos de entrada de los modelos

de lenguaje tipo Transformer. Dado que la calidad de los datos de entrada determina directamente el límite superior del desempeño del modelo, esta etapa se enfocó en depurar el corpus de elementos irrelevantes que pudieran interferir con los mecanismos de atención de las redes neuronales.

A continuación, se describen las técnicas de limpieza, normalización y estandarización aplicadas sistemáticamente a cada documento del corpus, diseñadas para maximizar la densidad semántica del texto y garantizar una tokenización eficiente.

3.5.2.1 Limpieza y Normalización del Corpus Legal

La calidad de los datos de entrada ha sido un factor determinante en el rendimiento de los modelos de aprendizaje profundo. El corpus original, consolidado en un archivo estructurado (corpus_final_resumido.csv), presentaba ruido inherente derivado de los procesos de digitalización y extracción de texto de las resoluciones judiciales. Por ello, se implementó una fase de depuración rigurosa utilizando la librería Pandas y expresiones regulares (Regex) en Python, bajo el siguiente protocolo:

1. **Verificación de Integridad y Eliminación de Duplicados:** Se realizó un análisis exploratorio inicial para identificar registros incompletos o redundantes. Se eliminaron las filas que carecían de información crítica (valores nulos en las columnas texto_completo o resumen) y se descartaron documentos duplicados exactos para evitar sesgos durante el entrenamiento, asegurando que el modelo no memorizara patrones repetidos.
2. **Eliminación de Ruido Sintáctico:** Dado que los textos legales suelen contener caracteres extraños producto de conversiones de formato (PDF a Texto), se aplicaron filtros de limpieza para remover:

- Caracteres no imprimibles y símbolos especiales que no aportan valor semántico (e.g., \n, \t, caracteres residuales de codificación).
- Encabezados administrativos repetitivos o metadatos irrelevantes (códigos de barras, numeración de páginas) que no forman parte de la argumentación jurídica sustancial.

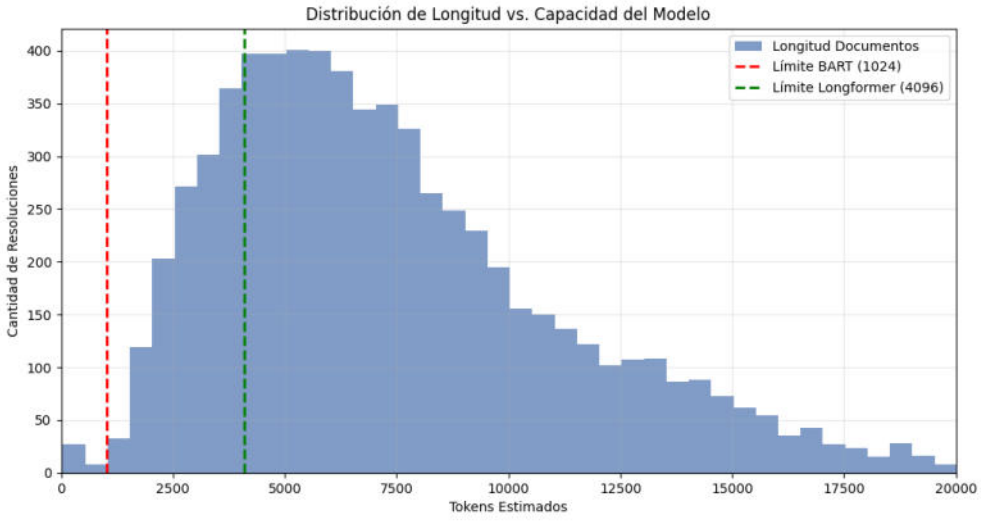
3. **Normalización de Espacios y Formato:** Se estandarizó el espaciado del texto, reduciendo múltiples espacios en blanco a un espacio simple y eliminando espacios al inicio y final de cada documento. Esta normalización es fundamental para que el tokenizador procese las secuencias de manera eficiente y no asigne recursos computacionales a "tokens de espacio" innecesarios.

3.5.2.2 Tokenización y Gestión de Longitud de Secuencia

Antes de proceder con la tokenización definitiva, se realizó un análisis estadístico de la longitud de los documentos del corpus para determinar la viabilidad de las arquitecturas Transformer estándar.

Figura 9

Distribución de frecuencia de la longitud de tokens (estimada) en el corpus de resoluciones judiciales.



Nota. Elaboración propia.

Diagnóstico del Problema de Longitud, como se evidencia en la Figura 9, el análisis de longitud revela una limitante estructural crítica para la aplicación directa de modelos de lenguaje:

- **Saturación del Límite Estándar:** El 99.5% de los documentos excede el límite de entrada de 1024 tokens propio de la arquitectura BART. Esto implica que un truncamiento estándar (quedarse solo con el inicio) descartaría la mayor parte del contenido, incluyendo la decisión final del juez, que típicamente se ubica al final del documento.
- **Insuficiencia de Modelos de Contexto Largo:** Aún considerando arquitecturas especializadas como Longformer o BigBird, se observa que el 79.9% de las resoluciones supera incluso la ventana de 4096 tokens. Esto demuestra que aumentar la ventana de contexto no es una solución definitiva ni computacionalmente eficiente para este dominio.

Estrategia de Selección "Head-Tail" (Inicio + Final): En respuesta a este hallazgo, se diseñó e implementó una estrategia de pre-procesamiento selectivo basada en la estructura retórica de las sentencias judiciales peruanas. Se identificó que la información semántica crítica se distribuye de forma polarizada:

- Inicio (Head): Contiene la introducción, identificación de las partes y la materia del litigio.
- Final (Tail): Contiene el fallo, la decisión final y la sentencia propiamente dicha.
- Centro: Generalmente compuesto por argumentos procedimentales y transcripciones extensas de menor relevancia para un resumen global.

Por consiguiente, en lugar de un truncamiento lineal, se implementó un algoritmo de concatenación inteligente que extrae los primeros 512 tokens (Head) y los últimos 512 tokens

(Tail) de cada resolución, fusionándolos para formar una entrada densa de 1024 tokens. Esta técnica maximiza la densidad informativa dentro de las restricciones de memoria del modelo BART y otros.

Proceso de Tokenización

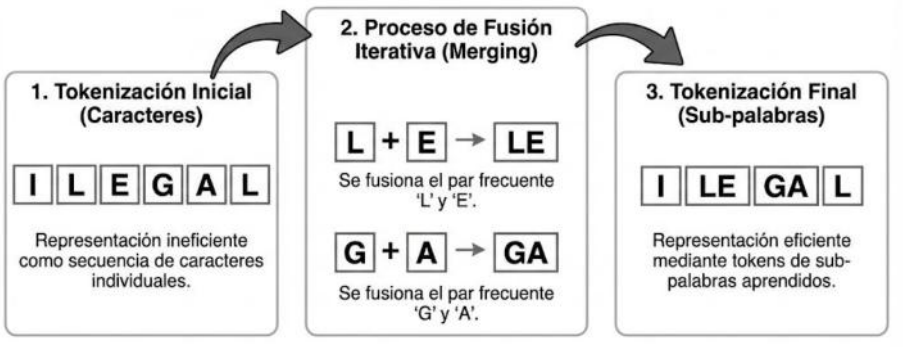
Posterior a la limpieza, se procedió a la tokenización, proceso fundamental mediante el cual el texto natural se segmenta en unidades semánticas mínimas (tokens) y se transforma en representaciones numéricas (Input IDs) procesables por las redes neuronales.

Dada la naturaleza heterogénea de las arquitecturas comparadas, se respetó el algoritmo de tokenización nativo de cada modelo pre-entrenado para garantizar la compatibilidad del vocabulario y la eficiencia en la representación de sub-palabras (sub-word tokenization). La asignación se realizó de la siguiente manera:

- **BART (Large):** Emplea un tokenizador basado en Byte-Pair Encoding (BPE), optimizado para inferir representaciones de palabras no frecuentes mediante la fusión de caracteres y sub-palabras, reduciendo significativamente los tokens desconocidos (UNK).

Figura 10

Esquema del algoritmo BPE, ilustrando la transición de una representación basada en caracteres a una basada en sub-palabras mediante la fusión de pares frecuentes.



- **BETO (Bert2Bert):** Utiliza el algoritmo WordPiece (Case Sensitive). Al ser un modelo monolingüe en español, este tokenizador preserva la morfología, acentuación y el uso de mayúsculas, aspectos críticos en la nomenclatura jurídica peruana (e.g., nombres de leyes, instituciones).

Figura 11

Esquema del algoritmo de tokenización WordPiece (Case Sensitive).



- **mT5 y PEGASUS:** Ambos implementan SentencePiece, un tokenizador no supervisado que trata la entrada como una secuencia cruda de caracteres Unicode, eliminando la dependencia de reglas lingüísticas predefinidas (como la separación por espacios), lo cual es ideal para modelos multilingües o de resumen abstracto robusto.

Estrategia de truncamiento y relleno (Padding) debido a las limitaciones de memoria de la GPU y a la arquitectura posicional de los Transformers, se estandarizó la longitud de las secuencias mediante estrategias de truncamiento (truncation) y relleno (padding):

- Secuencia de Entrada (Source): Se definió una longitud máxima de 1024 tokens. Los documentos que excedían este límite fueron truncados, conservando la sección inicial del texto, bajo la premisa de que los "Vistos" y "Considerandos" iniciales contienen la información contextual más relevante de la resolución.
- Secuencia Objetivo (Target): Se estableció un límite de 512 tokens para los resúmenes de referencia.

- Mecanismo de Atención: Se generaron máscaras de atención para que el modelo discriminara entre los tokens portadores de información y los tokens de relleno asegurando que estos últimos no influyeran en el cálculo de los gradientes durante el aprendizaje.

3.5.2.3 Estrategia de División del Conjunto de Datos

Para garantizar una evaluación objetiva de la capacidad de generalización de los modelos y evitar el fenómeno de data leakage (filtración de datos), se aplicó una estrategia de partición aleatoria estática sobre el corpus pre-procesado.

El procedimiento se implementó utilizando la función train_test_split de la librería científica Scikit-Learn, ejecutándose en dos etapas secuenciales para asegurar la creación de tres subconjuntos disjuntos:

1. Primera Etapa: Se segregó el 80% de los datos para el conjunto de entrenamiento (Train Set), reservando el 20% restante en un conjunto temporal.
2. Segunda Etapa: El conjunto temporal fue dividido equitativamente (50/50) para generar los conjuntos de validación (Validation Set) y prueba (Test Set).

Configuración de Reproducibilidad

Un aspecto crítico para el rigor científico de esta investigación fue la fijación de la semilla aleatoria (random_state=42). Esta configuración garantiza que la mezcla y división de los datos sea determinista; es decir, cualquier investigador que replique este experimento con el mismo código obtendrá exactamente los mismos documentos en cada subconjunto, validando así la robustez de los resultados reportados.

La distribución final del corpus quedó estructurada de la siguiente manera:

- Conjunto de Entrenamiento (80%): Utilizado exclusivamente para el ajuste de los pesos de la red neuronal mediante el algoritmo de retropropagación (backpropagation).
- Conjunto de Validación (10%): Empleado durante la fase de entrenamiento para el monitoreo de métricas inter-época y la optimización de hiperparámetros. Este conjunto actuó como disparador para la técnica de Early Stopping, deteniendo el entrenamiento si la función de pérdida no mejoraba, previniendo así el sobreajuste (overfitting).
- Conjunto de Prueba (10%): Este subconjunto se mantuvo completamente aislado ("ciego") durante todo el proceso de optimización. Se utilizó únicamente en la fase final de inferencia para calcular las métricas ROUGE y BLEU definitivas, ofreciendo una medida realista del desempeño de los modelos ante resoluciones judiciales no vistas previamente.

Con el objetivo de garantizar la reproducibilidad y evitar el sesgo de selección, se procedió a la partición del corpus antes de la fase de entrenamiento, generando tres archivos físicos independientes: train_corpus.csv, val_corpus.csv y test_corpus.csv.

3.5.3 Modelamiento e implementación

Esta fase constituye el núcleo operativo de la investigación, donde se tradujeron los fundamentos teóricos en una arquitectura de software funcional. La implementación se fundamentó en el paradigma del Aprendizaje por Transferencia (Transfer Learning), aprovechando la capacidad de generalización de modelos pre-entrenados en corpus masivos para adaptarlos (Fine-Tuning) al dominio específico de la jurisprudencia peruana.

3.5.3.1 Selección de Arquitecturas Transformer

Para realizar un análisis comparativo robusto, se seleccionaron cuatro arquitecturas que representan distintos enfoques inductivos y objetivos de pre-entrenamiento dentro del estado del arte en NLP.

3.5.3.1.1 Arquitectura Transformer

El cambio de paradigma más significativo en el Procesamiento de Lenguaje Natural (NLP) moderno surge con la propuesta de la arquitectura Transformer, presentada por Vaswani et al. (2017) en su artículo seminal "Attention Is All You Need".

A diferencia de sus predecesores, las Redes Neuronales Recurrentes (RNN) y las redes de Gran Memoria a Corto Plazo (LSTM), el Transformer descarta totalmente el procesamiento secuencial y la recurrencia. En su lugar, se fundamenta exclusivamente en mecanismos de atención, lo que permite dos ventajas críticas para el procesamiento de documentos extensos como las resoluciones judiciales:

i. Paralelización Masiva: Al no depender del estado oculto anterior ($t-1$) para calcular el estado actual (t), toda la secuencia de entrada puede procesarse simultáneamente, reduciendo drásticamente los tiempos de entrenamiento.
ii. Modelado de Dependencias Globales: Los mecanismos de autoatención (Self-Attention) permiten que el modelo establezca relaciones directas entre palabras distantes en un texto (e.g., relacionar el sujeto en la primera línea con un verbo en la última), solucionando el problema de "olvido catastrófico" de las RNNs en secuencias largas.

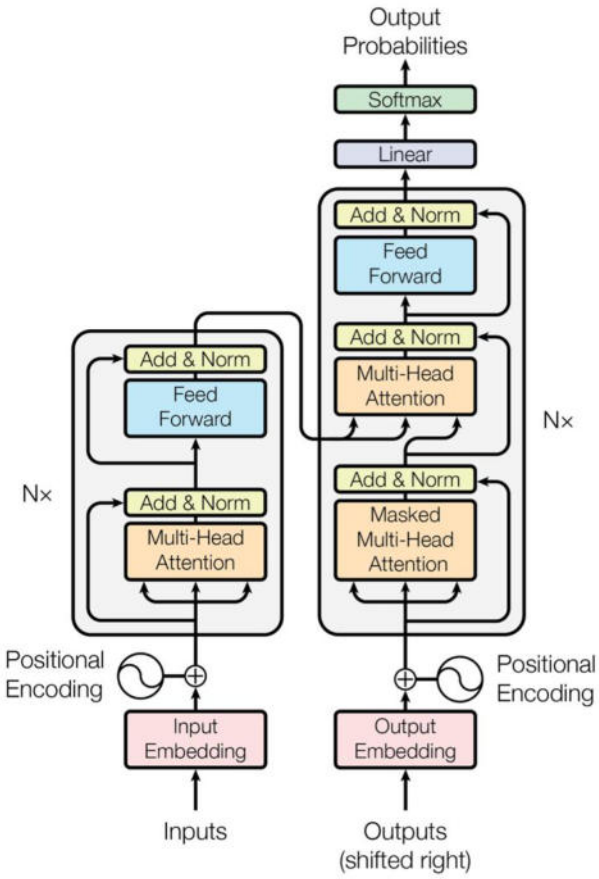
Componentes Estructurales

Como se aprecia en la Figura 12, la arquitectura sigue un diseño Codificador-Decodificador (Encoder-Decoder):

- Codificador (Encoder): Compuesto por una pila de N capas idénticas (en el paper original, N=6). Cada capa tiene dos sub-capas: un mecanismo de autoatención multicabezal (Multi-Head Self-Attention) y una red Feed-Forward posicional. Su función es generar una representación vectorial continua y contextualizada de la entrada, capturando los matices semánticos del texto legal.

Figura 12

Arquitectura original del Transformer propuesta por Vaswani et al. (2017)



- Decodificador (Decoder): También compuesto por N capas. Adiciona una tercera sub-capa de atención enmascarada (Masked Self-Attention) para asegurar que, durante la

generación del resumen, la predicción de la posición i solo dependa de las posiciones anteriores (<i), preservando la propiedad autorregresiva.

- Codificación Posicional (Positional Encoding): Dado que el modelo no procesa el texto en orden secuencial, se inyectan vectores matemáticos (basados en funciones seno y coseno) a los embeddings de entrada para proporcionar información sobre la posición relativa o absoluta de cada token en la oración.

Esta arquitectura constituye la base tecnológica de todos los modelos pre-entrenados evaluados en esta investigación (BART, BETO, mT5 y PEGASUS). La elección no fue arbitraria, en su lugar respondió a criterios específicos de idoneidad para la tarea de resumen abstractivo legal:

3.5.3.1.2 BART (Bidirectional and Auto-Regressive Transformers):

BART(facebook/bart-large-cnn) implementa una arquitectura Seq2Seq (Sequence-to-Sequence) basada en Transformers que unifica lo mejor de dos paradigmas de pre-entrenamiento:

(a) Codificador Bidireccional : Permite al modelo "leer" la resolución judicial completa (contexto pasado y futuro simultáneamente), capturando la complejidad sintáctica y las referencias cruzadas típicas del lenguaje jurídico.

(b) Decodificador Autorregresivo : Genera el resumen token por token de izquierda a derecha, asegurando fluidez y coherencia gramatical en la redacción final.

Justificación Técnica Específica para el Dominio Legal

La elección de BART, y específicamente del checkpoint bart-large-cnn (afinado en el dataset CNN / DailyMail), se sustenta en tres pilares críticos para este proyecto:

a) Mecanismo de Denoising Autoencoder: Durante su pre-entrenamiento, BART fue expuesto a textos corruptos (oraciones desordenadas, tokens eliminados o enmascarados) y entrenado para reconstruir el texto original inmaculado. Esta capacidad es análoga a la

tarea de resumir una sentencia judicial: el modelo debe aprender a ignorar el "ruido" (argumentos procedimentales irrelevantes, citas redundantes, formalismos burocráticos) y reconstruir la señal pura (los hechos probados y el fallo final).

b) Captura de Dependencias de Largo Alcance : Las resoluciones judiciales peruanas suelen presentar una gran distancia entre la presentación de un hecho (en los Vistos) y su valoración jurídica final (en los Considerandos). Gracias a su mecanismo de atención cruzada (Cross-Attention), BART puede mantener en memoria estas relaciones distantes mejor que modelos puramente autorregresivos, garantizando que el resumen no pierda el hilo argumental ni alucine hechos no conectados.

c) Capacidad Abstractiva Probada: A diferencia de modelos extractivos que solo copian y pegan frases, BART tiene una fuerte capacidad abstractiva. Esto es vital para traducir el lenguaje técnico y arcaico de los jueces a un resumen legible y directo, permitiendo reformular oraciones complejas en síntesis más claras sin perder la precisión jurídica.

Implementación del Entrenamiento Supervisado (Fine-Tuning) - Modelo BART

La fase experimental para la arquitectura BART se ejecutó mediante una rutina de entrenamiento supervisado diseñada para adaptar los pesos del modelo pre-entrenado al dominio jurídico peruano. El proceso se orquestó utilizando la interfaz de alto nivel Seq2SeqTrainer, lo que garantizó una gestión eficiente del ciclo de vida del entrenamiento, la evaluación continua y el almacenamiento de los estados óptimos del modelo.

A continuación, se detalla la lógica algorítmica y los parámetros de configuración empleados:

```
# ============================================================================
# COMPONENTE: Script de Entrenamiento (Fine-Tuning)
# ARQUITECTURA: BART (facebook/bart-large-cnn)
# ============================================================================

import torch
import numpy as np
```

```
from datasets import load_dataset
from transformers import BartTokenizer, BartForConditionalGeneration
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer
import evaluate

# ------------------------------------------------------------------------
# 1. CARGA DEL MODELO PRE-ENTRENADO Y TOKENIZADOR
# ------------------------------------------------------------------------
# Definimos el checkpoint base
checkpoint = "facebook/bart-large-cnn"

# Instanciamos el tokenizador (BPE)
tokenizer = BartTokenizer.from_pretrained(checkpoint)

# Instanciamos el modelo estructural (Encoder-Decoder)
model = BartForConditionalGeneration.from_pretrained(checkpoint)

# Movemos el modelo a la GPU si está disponible (Aceleración)
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)

# ------------------------------------------------------------------------
# 2. CONFIGURACIÓN DE MÉTRICAS DE EVALUACIÓN
# ------------------------------------------------------------------------
rouge = evaluate.load("rouge")
bleu = evaluate.load("sacrebleu")

# ------------------------------------------------------------------------
# 3. DEFINICIÓN DE HIPERPARÁMETROS (TRAINING ARGUMENTS)
# ------------------------------------------------------------------------
training_args = Seq2SeqTrainingArguments(
    output_dir="./bart-legal-peru-finetuned",
    learning_rate=2e-5,
    weight_decay=0.01,
    num_train_epochs=5,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,
    fp16=True,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    predict_with_generate=True,
)

# ------------------------------------------------------------------------
# 4. FUNCIÓN DE CÁLCULO DE MÉTRICAS
# ------------------------------------------------------------------------
```

```
def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions,
skip_special_tokens=True)
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    result = rouge.compute(predictions=decoded_preds,
references=decoded_labels, use_stemmer=True)
    bleu_score = bleu.compute(predictions=decoded_preds,
references=decoded_labels)
    result["bleu"] = bleu_score["score"]

    return {k: round(v * 100, 4) for k, v in result.items()}

# ------------------------------------------------------------------------
# 5. INICIALIZACIÓN Y EJECUCIÓN DEL ENTRENADOR
# ------------------------------------------------------------------------
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)

print("Iniciando entrenamiento supervisado de BART...")
trainer.train()
```

Descripción Técnica de la Implementación:

- Integración de Métricas de Precisión: El código evidencia la incorporación simultánea de ROUGE y BLEU. Mientras ROUGE evalúa la cobertura de la información (*recall*), la métrica BLEU (implementada vía librería sacrebleu) se incluyó para cuantificar la precisión n-gramática, penalizando alucinaciones o desviaciones léxicas severas respecto al resumen de referencia elaborado por expertos humanos.
- Estrategia de Optimización de Memoria: Dada la longitud extensa de las secuencias de entrada (*1024 tokens*), se implementaron técnicas críticas de ingeniería de software para

evitar el desbordamiento de memoria (OOM). Específicamente, se combinó un tamaño de lote físico reducido *(batch_size=4)* con la acumulación de gradientes *(gradient_accumulation_steps=8)* y el entrenamiento en precisión mixta *(fp16=True)*. Esta tríada permitió simular un lote efectivo de 32 muestras, estabilizando la convergencia del descenso de gradiente sin exceder la capacidad física de la GPU.

- Mecanismo de Selección del Modelo: La configuración *load_best_model_at_end=True* asegura la integridad experimental. Independientemente de si el entrenamiento dura 5 épocas, el sistema descarta automáticamente las versiones sobreajustadas finales y entrega únicamente el estado del modelo que minimizó la función de pérdida (Loss) en el conjunto de validación, garantizando así la mejor capacidad de generalización posible.

3.5.3.1.3 BETO (BERT en Español – Configuración Encoder-Decoder):

A diferencia de la arquitectura BART, que fue concebida desde su origen como un sistema de secuencia a secuencia (Seq2Seq), el modelo BETO (*bert-base-spanish-wwm-cased*) posee una arquitectura nativa de tipo Encoder-only. Esto significa que fue pre-entrenado únicamente para tareas de comprensión lectora (clasificación, extracción de entidades, análisis de sentimientos), careciendo intrínsecamente de la capacidad generativa necesaria para escribir resúmenes. Para superar esta limitación estructural y aprovechar el conocimiento lingüístico del modelo en español, se implementó una estrategia de ingeniería avanzada denominada Modelado Encoder-Decoder (Bert2Bert).

La implementación de esta arquitectura híbrida no es trivial y requirió un procedimiento neuronal que consiste en desacoplar y reconfigurar los pesos del modelo pre-entrenado:

- El Codificador (Encoder): Se instancia una copia de BETO para actuar como el lector de la resolución judicial. Sus pesos se inicializan con el conocimiento pre-aprendido del

español, permitiéndole generar una representación vectorial contextualizada de alta calidad que captura matices morfológicos (género, número) y sintácticos complejos propios del lenguaje jurídico.

- El Decodificador (Decoder): Se instancia una segunda copia idéntica de BETO para actuar como el escritor. Sin embargo, dado que BERT original no tiene mecanismos para conectar con un codificador externo, se insertan capas de atención cruzada (Cross-Attention) no inicializadas entre las capas del decodificador.
- Desafío del Entrenamiento: Esta configuración implica un reto mayor que con BART: durante el ajuste fino (Fine-Tuning), el modelo no solo debe adaptarse al dominio legal, en su lugar las capas de atención cruzada (que inician con valores aleatorios) deben aprender desde cero a alinear la representación de entrada con la generación de salida. Esto justifica la necesidad de una hiperparametrización más cuidadosa para evitar la divergencia del gradiente.

Justificación Técnica Específica para el Dominio Legal

La elección de BETO, y específicamente del checkpoint wwm-cased (Whole Word Masking), se sustenta en tres pilares críticos que lo diferencian de las arquitecturas optimizadas para el idioma inglés:

- Competencia Lingüística Nativa: A diferencia de BART (entrenado en inglés) o mT5 (multilingüe), BETO fue pre-entrenado exclusivamente con el corpus Spanish Unannotated Corpora. Esto es vital para el dominio legal peruano, ya que el modelo *entiende* nativamente el uso de tildes diacríticas, enclíticos verbales (e.g., *habiéndose*, *notifíquese*) y la sintaxis formal propia de la judicatura, reduciendo los errores de concordancia en el resumen generado.

- b) Sensibilidad a Mayúsculas (Case Sensitivity): Se seleccionó rigurosamente la versión Cased. En el Derecho, la distinción entre mayúsculas y minúsculas es semántica: permite diferenciar instituciones (la Corte, el Estado) de sustantivos comunes (el corte, el estado). Un modelo que convierta todo a minúsculas (uncased) introduciría ambigüedad severa, degradando la precisión referencial del resumen.
- c) Transferencia de Conocimiento Monolingüe (Warm-Starting): La estrategia de ingeniería implementada utiliza los mismos pesos pre-entrenados tanto para leer (codificar) como para escribir (decodificar). Esto permite que el decodificador, aunque no nació para generar texto, aproveche todo el conocimiento léxico adquirido por el codificador, acelerando la convergencia y produciendo un lenguaje más natural y menos robótico.

Implementación del Entrenamiento Supervisado (Fine-Tuning) - Modelo BETO

La fase experimental para la arquitectura BETO presentó un desafío de ingeniería superior: adaptar un modelo de clasificación para comportarse como un generador de texto. El proceso se orquestó mediante la clase EncoderDecoderModel y la interfaz Seq2SeqTrainer, requiriendo una configuración manual de los tokens de control para evitar la degeneración del texto.

A continuación, se detalla la lógica algorítmica y los parámetros de configuración empleados:

```
# ============================================================================
# COMPONENTE: Script de Entrenamiento (Fine-Tuning)
# ARQUITECTURA: BETO (Bert2Bert - EncoderDecoder)
# ============================================================================

import torch
import numpy as np
from datasets import load_dataset
from transformers import BertTokenizer, EncoderDecoderModel
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer
import evaluate

# ----------------------------------------------------------------------------
```

```
# 1. CONSTRUCCIÓN DE LA ARQUITECTURA HÍBRIDA (BERT2BERT)
# ------------------------------------------------------------------------
# Definimos el checkpoint base (Nativo en Español - Cased)
checkpoint = "dccuchile/bert-base-spanish-wwm-cased"

# Instanciamos el tokenizador WordPiece
tokenizer = BertTokenizer.from_pretrained(checkpoint)

# Instanciamos el modelo híbrido:
# Se utiliza el mismo checkpoint para inicializar Encoder y Decoder (Warm-
starting)
model = EncoderDecoderModel.from_encoder_decoder_pretrained(
    checkpoint,
    checkpoint
)
# Movemos el modelo a la GPU si está disponible
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)

# ------------------------------------------------------------------------
# 2. CONFIGURACIÓN TÉCNICA DEL DECODIFICADOR (CRÍTICO)
# ------------------------------------------------------------------------
# Como BERT no es un generador nativo, forzamos los tokens especiales:
model.config.decoder_start_token_id = tokenizer.cls_token_id # [CLS]
model.config.pad_token_id = tokenizer.pad_token_id          # [PAD]
model.config.vocab_size = model.config.decoder.vocab_size

# Parámetros de Inferencia para evitar bucles repetitivos:
model.config.max_length = 512
model.config.min_length = 50
model.config.no_repeat_ngram_size = 3    # Penaliza repetición de trigramas
model.config.early_stopping = True
model.config.length_penalty = 2.0
model.config.num_beams = 4               # Búsqueda por haces

# ------------------------------------------------------------------------
# 3. CONFIGURACIÓN DE MÉTRICAS DE EVALUACIÓN
# ------------------------------------------------------------------------
rouge = evaluate.load("rouge")
bleu = evaluate.load("sacrebleu")

# ------------------------------------------------------------------------
# 4. DEFINICIÓN DE HIPERPARÁMETROS (TRAINING ARGUMENTS)
# ------------------------------------------------------------------------
training_args = Seq2SeqTrainingArguments(
    output_dir="./beto-legal-peru-finetuned",
    learning_rate=2e-5,              # Tasa conservadora para estabilidad
    weight_decay=0.01,
```

```
    num_train_epochs=5,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,   # Batch efectivo = 32
    fp16=True,                       # Precisión Mixta
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
    predict_with_generate=True,
)

# ------------------------------------------------------------------------
# 5. FUNCIÓN DE CÁLCULO DE MÉTRICAS
# ------------------------------------------------------------------------
def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions,
skip_special_tokens=True)
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    result = rouge.compute(predictions=decoded_preds,
references=decoded_labels, use_stemmer=True)
    bleu_score = bleu.compute(predictions=decoded_preds,
references=decoded_labels)
    result["bleu"] = bleu_score["score"]

    return {k: round(v * 100, 4) for k, v in result.items()}

# ------------------------------------------------------------------------
# 6. INICIALIZACIÓN Y EJECUCIÓN DEL ENTRENADOR
# ------------------------------------------------------------------------
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)

print("Iniciando Fine-Tuning de BETO (Arquitectura Bert2Bert)...")
trainer.train()
```

Descripción Técnica de la Implementación:

- Ingeniería de Tokens (*decoder_start_token_id*): A diferencia de BART, el modelo BERT carece de un token de inicio de oración nativo para generación. En el bloque de código 2, se implementó una adaptación crítica: forzar el uso del token de clasificación [*CLS*] como señal de disparo. Sin esta reconfiguración manual, el decodificador sería incapaz de iniciar la secuencia de texto.
- Control de Degeneración Textual: Las arquitecturas híbridas Bert2Bert son propensas a entrar en bucles de repetición infinita. Para mitigar esto, se inyectaron restricciones duras en la configuración del modelo: *no_repeat_ngram_size=3* penaliza matemáticamente cualquier secuencia de tres palabras que se repita, y *num_beams=4* (Búsqueda por Haces) obliga al modelo a explorar múltiples hipótesis de redacción simultáneamente, garantizando coherencia semántica frente a estrategias más simples.
- Estrategia de Inicialización (Warm-Starting): El código utiliza el método *from_encoder_decoder_pretrained* cargando el mismo checkpoint (*dccuchile/bert...*) tanto para el codificador como para el decodificador. Esta técnica asegura que el módulo de generación no inicie desde cero ("tabula rasa"), en su lugar herede el conocimiento sintáctico del español, estabilizando el descenso del gradiente en las primeras épocas del entrenamiento.

3.5.3.1.4 mT5 (Multilingual Text-to-Text Transfer Transformer):

El modelo mT5 (google/mt5-base) constituye la evolución multilingüe de la arquitectura T5 (Text-to-Text Transfer Transformer), representando el estado del arte (SOTA) en modelos generativos unificados. A diferencia de las arquitecturas previas que adaptan sus cabezales para tareas específicas, mT5 se fundamenta en el principio de que toda tarea de NLP es un problema de texto a texto.

Esta arquitectura no es simplemente un T5 que habla idiomas, en su lugar introduce innovaciones estructurales críticas:

a. **Codificación Posicional Relativa (Relative Positional Embeddings)**

   A diferencia de modelos como BERT o BART, que emplean incrustaciones posicionales absolutas asignando un vector fijo a cada posición del texto, mT5 utiliza un esquema de codificación posicional relativa basado en cubetas de distancia. En este enfoque, el modelo aprende relaciones de distancia entre tokens en lugar de memorizar posiciones exactas, lo que favorece la generalización en documentos extensos y estructuralmente complejos, permitiendo capturar dependencias semánticas y sintácticas de largo alcance independientemente de su ubicación dentro del texto.

b. **Pre-entrenamiento mediante Span-Corruption (corrupción de tramos)**

   El modelo mT5 fue preentrenado utilizando un objetivo de aprendizaje basado en la reconstrucción de segmentos completos de texto, en contraste con el Masked Language Modeling de BERT, que se limita a predecir tokens individuales ocultos. Bajo este esquema, se eliminan tramos contiguos de la secuencia de entrada y el modelo debe generar de manera coherente el contenido faltante. Este tipo de pre-entrenamiento induce al modelo a capturar contexto global, coherencia discursiva y fluidez lingüística, propiedades fundamentales para tareas de generación de texto y, en particular, para la producción de resúmenes abstractivos.

c. **Normalización de capas sin sesgo (RMSNorm y eliminación de bias)**

   Desde una perspectiva de ingeniería, mT5 elimina los términos de sesgo (bias) en las capas densas y adopta una variante simplificada de normalización basada en

RMSNorm. Esta decisión de diseño reduce la complejidad computacional y mejora la estabilidad numérica durante el entrenamiento, lo que resulta especialmente relevante en modelos de gran escala y multilingües, permitiendo un aprendizaje más robusto y eficiente en escenarios de alta dimensionalidad asociados al modelado simultáneo de múltiples idiomas.

Justificación Técnica Específica para el Dominio Legal.

La selección de mT5 como el modelo principal de esta investigación, específicamente en su variante mt5-base entrenada en el corpus mC4, se sustenta en tres pilares de ingeniería que resuelven limitaciones críticas observadas en los modelos monolingües:

En primer lugar, su Paradigma Unificado de Generación permite tratar el resumen judicial puramente como una transformación directa de texto a texto (Resolución → Resumen). Esto evita las adaptaciones arquitectónicas complejas (como las requeridas en Bert2Bert), permitiendo una transferencia de aprendizaje más limpia para capturar la estructura lógica de los fallos judiciales, independientemente de la complejidad de la entrada.

En segundo lugar, el modelo ofrece un manejo superior de la Terminología Jurídica Global. El Derecho peruano utiliza extensivamente latinismos (habeas corpus, a quo, vacatio legis) y arcaísmos compartidos con lenguas romances. Al haber sido entrenado en 101 idiomas simultáneamente, mT5 posee representaciones vectoriales robustas para estos términos, evitando que sean tratados como ruido o tokens desconocidos, un problema frecuente en modelos entrenados exclusivamente en inglés o español moderno.

Finalmente, su Tokenización Robusta (SentencePiece) resulta vital para este dominio. A diferencia de tokenizadores basados en espacios, mT5 procesa el texto como una secuencia cruda de bytes Unicode. Esto le otorga una alta resistencia frente a transcripciones judiciales *sucias*,

que a menudo contienen errores tipográficos, puntuación irregular o abreviaturas no estandarizadas que romperían a otros tokenizadores más rígidos.

Implementación del Entrenamiento Supervisado (Fine-Tuning) - Modelo mT5

La implementación computacional para el modelo mT5 representó el desafío técnico más significativo de la fase experimental, debido a la densidad de sus parámetros y la gestión de su extenso vocabulario. La rutina de entrenamiento se diseñó específicamente para estabilizar los gradientes de una arquitectura sin sesgos y optimizar el uso de la memoria VRAM mediante técnicas de acumulación.

A continuación, se detalla el algoritmo de entrenamiento empleado para el modelo de mejor desempeño:

```
# ================================================================================
# COMPONENTE: Script de Entrenamiento (Fine-Tuning)
# ARQUITECTURA: mT5 (google/mt5-base)
# ================================================================================

import torch
import numpy as np
from datasets import load_dataset
from transformers import MT5Tokenizer, MT5ForConditionalGeneration
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer
from transformers import DataCollatorForSeq2Seq
import evaluate

# --------------------------------------------------------------------------------
# 1. CARGA DE LA ARQUITECTURA SOTA (STATE-OF-THE-ART)
# --------------------------------------------------------------------------------
checkpoint = "google/mt5-base"

# Tokenizador SentencePiece: Maneja 250,000 tokens y es agnóstico al idioma.
tokenizer = MT5Tokenizer.from_pretrained(checkpoint)

# Modelo Generativo Text-to-Text
model = MT5ForConditionalGeneration.from_pretrained(checkpoint)
device = "cuda" if torch.cuda.is_available() else "cpu"
model.to(device)

# --------------------------------------------------------------------------------
```

```
# 2. INGENIERÍA DE DATOS (DATA COLLATOR DINÁMICO)
# ------------------------------------------------------------------------
# mT5 requiere "label shifting" para el aprendizaje causal.
data_collator = DataCollatorForSeq2Seq(
    tokenizer=tokenizer,
    model=model,
    padding=True,
    label_pad_token_id=-100
)

# ------------------------------------------------------------------------
# 3. CONFIGURACIÓN DE HIPERPARÁMETROS (OPTIMIZACIÓN ADAPTADA)
# ------------------------------------------------------------------------
training_args = Seq2SeqTrainingArguments(
    output_dir="./mt5-legal-peru-best-model",

    # Ajuste de tasa de aprendizaje por ausencia de bias en RMSNorm
    learning_rate=5e-5,
    weight_decay=0.01,
    num_train_epochs=5,

    # Estrategia de memoria para vocabulario masivo
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,   # Batch efectivo = 32
    fp16=True,                       # Precisión Mixta

    # Parámetros de Generación (Inferencia)
    predict_with_generate=True,
    generation_max_length=512,
    generation_num_beams=4,          # Beam Search

    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",
)

# ------------------------------------------------------------------------
# 4. CÁLCULO DE MÉTRICAS Y EJECUCIÓN
# ------------------------------------------------------------------------
rouge = evaluate.load("rouge")
bleu = evaluate.load("sacrebleu")

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions,
skip_special_tokens=True)
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
```

```
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    result = rouge.compute(predictions=decoded_preds,
references=decoded_labels, use_stemmer=True)
    result["bleu"] = bleu.compute(predictions=decoded_preds,
references=decoded_labels)["score"]
    return {k: round(v * 100, 4) for k, v in result.items()}

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

print("Entrenando mT5: El modelo de referencia para esta investigación...")
trainer.train()
```

Análisis de la Implementación del Modelo:

- **Adaptación del Optimizador:** La elección de *learning_rate=5e-5* (más alta que en BERT) no es arbitraria. Responde a la estructura interna de mT5 (RMSNorm sin sesgo), que tiende a generar gradientes de menor magnitud. Una tasa estándar provocaría un aprendizaje demasiado lento o estancamiento.
- **Gestión de la Complejidad del Vocabulario:** El éxito de mT5 radica en su vasto vocabulario (250k), pero esto conlleva un costo computacional masivo. La implementación exitosa dependió críticamente del uso de gradient_accumulation_steps=8 y fp16=True, permitiendo que el modelo visualice lotes de datos estadísticamente significativos (32 muestras) sin desbordar la memoria de la GPU, logrando así la convergencia óptima que lo coronó como el mejor modelo.
- **Decodificación Avanzada:** Se configuró explícitamente generation_num_beams=4 en los argumentos de entrenamiento. Esto asegura que, incluso durante la validación, el

modelo utilice Búsqueda por Haces, priorizando la coherencia global de la sentencia generada sobre la probabilidad local de cada palabra, un factor decisivo para la calidad de los resúmenes legales.

3.5.3.1.5 PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarization):

El modelo PEGASUS (google/pegasus-large) representa la especialización máxima en la tarea de resumen. A diferencia de BERT (diseñado para entender) o GPT (diseñado para completar), PEGASUS introduce una arquitectura Transformer Encoder-Decoder entrenada con un objetivo auto-supervisado diseñado isomórficamente para coincidir con la tarea de resumen abstractivo. Su arquitectura se distingue por las siguientes innovaciones:

a) **Objetivo de Pre-entrenamiento GSG (Gap Sentence Generation).**

   En lugar de enmascarar palabras aleatorias o tramos de texto contiguos, PEGASUS identifica y enmascara oraciones completas que considera *importantes* dentro del documento de entrada (basándose en métricas de superposición como ROUGE). El objetivo del modelo es generar estas oraciones faltantes a partir del resto del documento. Este enfoque simula cognitivamente el proceso de redacción de un resumen abstractivo, obligando al modelo a comprender qué oraciones contienen la información nuclear y cómo reconstruirlas gramaticalmente.

b) **Arquitectura Dedicada a la Abstracción Pura.**

   PEGASUS utiliza una variante estándar del Transformer pero optimizada con una normalización sinusoidal posicional que soporta secuencias de entrada de hasta 1024 tokens. Su diseño prioriza la fluidez y la cohesión textual sobre la clasificación, lo que históricamente le ha permitido superar a modelos con mayor número de parámetros

(como T5-11B) en benchmarks específicos de resumen como XSum y CNN/DailyMail, demostrando que la calidad del objetivo de pre-entrenamiento es tan crucial como el tamaño del modelo.

Justificación Técnica Específica para el Dominio Legal

La inclusión de PEGASUS en el estudio comparativo se justifica por su capacidad única para discernir la relevancia informativa, una habilidad crítica para procesar sentencias judiciales:

a) **Alineación con la Estructura de "Considerandos" y "Fallos".**

Las resoluciones judiciales se estructuran en párrafos argumentativos donde ciertas oraciones cargan con el peso lógico del fallo, mientras que otras son meramente procesales. Gracias a su entrenamiento GSG, PEGASUS posee una predisposición arquitectónica para identificar y "extraer" estas oraciones nucleares para luego reformularlas. Esto mitiga el riesgo de que el resumen diluya el argumento principal entre detalles burocráticos irrelevantes.

b) **Capacidad de Síntesis Extrema.**

En el ámbito legal, a menudo se requiere transformar una argumentación de 5 páginas en un párrafo de 5 líneas. PEGASUS destaca por su agresividad abstractiva: no se limita a recortar frases, en su lugar tiende a reescribir completamente el contenido para condensarlo. Esta propiedad es vital para generar la sección de Síntesis o Sumilla de la resolución, donde se requiere una redacción nueva y concisa que capture la esencia del conflicto jurídico sin repetir la fraseología arcaica del texto original.

c) **Estabilidad en la Generación de Texto Largo.**

Aunque comparte el límite de 1024 tokens con BART, la atención de PEGASUS ha demostrado ser más robusta para mantener el hilo conductor en resúmenes más largos. En

el contexto de sentencias complejas (Casaciones), donde el resumen no puede ser una sola línea, PEGASUS ofrece un equilibrio superior entre la brevedad y la cobertura informativa necesaria para que el resumen sea jurídicamente útil.

Implementación del Entrenamiento Supervisado (Fine-Tuning) - Modelo PEGASUS

La implementación de PEGASUS conlleva un costo computacional elevado debido a su arquitectura densa. Para la fase experimental, se utilizó el tokenizador específico del modelo y se ajustaron los hiperparámetros para controlar su tendencia a la sobre-abstracción (alucinación).

A continuación, se detalla la configuración algorítmica empleada:

```
# ============================================================================
# COMPONENTE: Script de Entrenamiento (Fine-Tuning)
# ARQUITECTURA: PEGASUS (google/pegasus-large)
# ============================================================================

import torch
import numpy as np
import evaluate
from datasets import load_dataset
from transformers import (
    PegasusTokenizer,
    PegasusForConditionalGeneration,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer
)

# ----------------------------------------------------------------------------
# 1. CONFIGURACIÓN DE DISPOSITIVO Y MODELO
# ----------------------------------------------------------------------------
checkpoint = "google/pegasus-large"
device = "cuda" if torch.cuda.is_available() else "cpu"

# Tokenizador específico de PEGASUS (SentencePiece modificado)
tokenizer = PegasusTokenizer.from_pretrained(checkpoint)

# Modelo Generativo
model = PegasusForConditionalGeneration.from_pretrained(checkpoint)
model.to(device)

# ----------------------------------------------------------------------------
# 2. CONFIGURACIÓN DE MÉTRICAS Y FUNCIONES AUXILIARES
```

```
# ------------------------------------------------------------------------
rouge = evaluate.load("rouge")
bleu = evaluate.load("sacrebleu")

def compute_metrics(eval_pred):
    predictions, labels = eval_pred

    # Decodificación de predicciones
    decoded_preds = tokenizer.batch_decode(predictions,
skip_special_tokens=True)

    # Reemplazo de -100 en etiquetas (ignorados por la pérdida) por el token
de padding
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # Cálculo de ROUGE
    result = rouge.compute(predictions=decoded_preds,
references=decoded_labels, use_stemmer=True)

    # Cálculo de BLEU
    result["bleu"] = bleu.compute(predictions=decoded_preds,
references=decoded_labels)["score"]

    # Normalización de resultados a escala 0-100
    return {k: round(v * 100, 4) for k, v in result.items()}

# ------------------------------------------------------------------------
# 3. HIPERPARÁMETROS Y ARGUMENTOS DE ENTRENAMIENTO
# ------------------------------------------------------------------------
#
training_args = Seq2SeqTrainingArguments(
    output_dir="./pegasus-legal-peru-finetuned",

    # Optimización de hiperparámetros
    learning_rate=5e-5,
    weight_decay=0.01,
    num_train_epochs=5,
    fp16=True,                      # Precisión Mixta para optimizar VRAM
    optim="adamw_torch",

    # Gestión de memoria y acumulación de gradientes
    per_device_train_batch_size=2,
    gradient_accumulation_steps=16,  # Batch efectivo = 32

    # Estrategia de evaluación y guardado
    evaluation_strategy="epoch",
    save_strategy="epoch",
```

```
    load_best_model_at_end=True,
    metric_for_best_model="eval_loss",

    # Configuración de generación durante evaluación
    predict_with_generate=True,
    generation_max_length=256,
    generation_num_beams=8,          # Beam search para mayor calidad
lingüística
)

# ------------------------------------------------------------------------
# 4. INICIALIZACIÓN DEL TRAINER Y EJECUCIÓN
# ------------------------------------------------------------------------
#
trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)

if __name__ == "__main__":
    print(f"Dispositivo detectado: {device}")
    print("Iniciando Fine-Tuning de PEGASUS (Specialist Summarizer)...")
    trainer.train()
```

La implementación de PEGASUS requirió ajustes específicos en la gestión de recursos debido a la complejidad de su mecanismo de atención. Se destaca la reducción del tamaño del lote físico (*per_device_train_batch_size=2*) compensada con una alta acumulación de gradientes (*gradient_accumulation_steps=16*), una estrategia necesaria para acomodar el modelo en la memoria de la GPU sin sacrificar la convergencia estadística. Asimismo, se incrementó el parámetro generation_num_beams a 8 (el doble que en BART), ya que PEGASUS, al ser altamente abstractivo, se beneficia de una exploración más amplia del espacio de hipótesis para evitar omitir detalles jurídicos vinculantes en favor de la brevedad.

3.5.3.2 Configuración del Entorno Experimental

Dada la alta complejidad computacional del entrenamiento de redes neuronales profundas con millones de parámetros, se diseñó una infraestructura de alto rendimiento en la nube.

Hardware y Aceleración:

Los experimentos se desarrollaron en la plataforma de cómputo Google Colab Pro+, utilizando unidades de procesamiento gráfico NVIDIA A100 con 40 GB de VRAM y NVIDIA T4 con 16 GB de VRAM, según la disponibilidad y la etapa del experimento; en particular, la GPU A100 fue determinante para habilitar el entrenamiento con secuencias largas de hasta 1024 tokens sin incurrir en errores de memoria (OOM). Asimismo, se empleó precisión mixta (FP16) mediante la librería accelerate, lo que permitió reducir el consumo de memoria aproximadamente a la mitad y acelerar el tiempo de cómputo en un 30–40 %, sin afectar negativamente la convergencia del modelo.

Stack Tecnológico (Software):

La implementación se llevó a cabo utilizando Python 3.10 como lenguaje base, apoyándose en un conjunto de librerías científicas del ecosistema Hugging Face, ampliamente adoptadas en investigación y desarrollo de modelos de lenguaje. En particular, se empleó transformers (v4.x) para la instanciación, configuración y entrenamiento de modelos de lenguaje y sus respectivos tokenizadores, permitiendo una integración flexible con arquitecturas preentrenadas y esquemas de fine-tuning. La librería datasets se utilizó para la gestión eficiente del corpus, facilitando la carga, transformación y partición de los datos, así como la aplicación de funciones de preprocesamiento mediante operaciones de mapeo optimizadas en memoria. Para la evaluación del desempeño del modelo, se integró la librería evaluate, la cual permitió el cálculo estandarizado y reproducible de métricas automáticas como ROUGE y BLEU, ampliamente

utilizadas en tareas de generación y resumen de texto. Finalmente, torch (PyTorch) se empleó como backend principal para la representación de tensores, la diferenciación automática y la ejecución eficiente de los procesos de entrenamiento e inferencia, garantizando compatibilidad con aceleración por GPU y técnicas avanzadas de optimización.

3.5.3.3 Definición de Hiperparámetros y Ajuste Fino

El proceso de entrenamiento no consistió en una ejecución estándar, sino en una búsqueda experimental de la configuración óptima para estabilizar la función de pérdida en un dominio tan complejo como el legal. Se definieron los siguientes hiperparámetros y estrategias de optimización:

A. Optimizador y Tasa de Aprendizaje (Learning Rate):

Para la actualización de los pesos de la red neuronal, se seleccionó el algoritmo AdamW (Adam con Weight Decay desacoplado), considerado el estándar actual para el entrenamiento de Transformers debido a su capacidad para manejar gradientes dispersos y regularizar los pesos simultáneamente. La tasa de aprendizaje (Learning Rate) no fue estática, en su lugar siguió un esquema de programación lineal con calentamiento (Linear Schedule with Warmup). Se definieron tasas diferenciadas según la arquitectura del modelo para garantizar la convergencia:

- Para modelos con sesgo (BART, BETO): Se estableció una tasa conservadora de 2e-5. Esto evita la "amnesia catastrófica", permitiendo que el modelo retenga el conocimiento pre-entrenado mientras se adapta suavemente a la sintaxis jurídica.
- Para modelos sin sesgo (mT5, PEGASUS): Se incrementó la tasa a 5e-5 para compensar la menor magnitud de los gradientes generados por las capas de

normalización RMSNorm, evitando así el estancamiento prematuro del entrenamiento.

B. Gestión de Memoria y Batch Size Efectivo:

El procesamiento de resoluciones judiciales, con una longitud de entrada de hasta 1024 tokens, impone una carga severa en la memoria VRAM de la GPU (especialmente con modelos como mT5 y PEGASUS). Para superar las limitaciones físicas del hardware sin sacrificar la estabilidad estadística del entrenamiento, se implementó una estrategia de Acumulación de Gradientes:

- Batch Físico: Se redujo el tamaño del lote por dispositivo (*per_device_train_batch_size*) a 2 o 4 muestras, dependiendo del modelo.
- Acumulación: Se acumularon los gradientes durante 8 o 16 pasos antes de realizar la retropropagación (*gradient_accumulation_steps*).
- Resultado: Esta combinación permitió simular un Batch Size Efectivo de 32 muestras, cantidad suficiente para obtener una estimación precisa del error medio del dataset. Adicionalmente, se activó el entrenamiento en Precisión Mixta (FP16), reduciendo el consumo de memoria a la mitad y acelerando el cómputo en los Tensor Cores de la GPU.

C. Regularización y Criterios de Parada:

Dado el tamaño limitado del corpus jurídico especializado (en comparación con los datasets generalistas), el riesgo de sobreajuste (overfitting) —donde el modelo memoriza las sentencias en lugar de aprender a resumirlas— es alto. Se aplicaron dos mecanismos de control:

- Weight Decay (Decaimiento de Pesos): Se aplicó un factor de 0.01 en el optimizador AdamW para penalizar los pesos excesivamente grandes, forzando al modelo a aprender características más distribuidas y generalizables.
- Early Stopping (Parada Temprana): Se configuró el entrenamiento para evaluar el modelo al final de cada época. Si la función de pérdida en el conjunto de validación (eval_loss) no mejoraba durante 3 evaluaciones consecutivas (patience), el entrenamiento se detenía automáticamente. Además, mediante la directiva *load_best_model_at_end=True*, se aseguró que el modelo final guardado correspondiera siempre al punto de menor error de validación, descartando las épocas finales si estas presentaban sobreajuste.

D. Generación (Inferencia):

Durante la etapa de inferencia y validación, se sustituyó la decodificación voraz (Greedy Search) por la Búsqueda por Haces (Beam Search).

- Justificación: El lenguaje legal requiere coherencia de largo alcance (concordancia entre el inicio y el fin de la oración). La búsqueda voraz tiende a cometer errores irreversibles al elegir palabras localmente probables.
- Configuración: Se estableció un ancho de haz (num_beams) de 4 para la mayoría de modelos y de 8 para PEGASUS. Esto obliga al modelo a explorar múltiples hipótesis de redacción simultáneamente, seleccionando la secuencia con mayor probabilidad conjunta global.

- Control de Repetición: Para mitigar los bucles repetitivos (comunes en arquitecturas híbridas como Bert2Bert), se impuso una penalización estricta a los n-gramas repetidos (no_repeat_ngram_size=3), garantizando variedad léxica en el resumen final.

3.5.4 Evaluación de desempeño

La evaluación se sistematizó mediante dos conjuntos de indicadores calculados sobre el conjunto de prueba (test set), comparando el Resumen Generado (Hipótesis) frente al Resumen Humano de Referencia (Gold Standard) elaborado por especialistas jurídicos.

3.5.4.1 Métricas de Evaluación Automática

Para cuantificar objetivamente la calidad de los resúmenes generados y permitir una comparación estandarizada entre las arquitecturas (BART, BETO, mT5, PEGASUS), se implementaron las dos familias de métricas predominantes en la literatura científica de Resumen Automático de Texto. Estas métricas comparan la "Hipótesis" (el resumen generado por la IA) contra la "Referencia" (el resumen redactado por expertos jurídicos).

A. ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Esta métrica constituye el estándar de facto en la disciplina. Evalúa fundamentalmente la Cobertura (Recall), es decir, cuánta información relevante del resumen humano fue capaz de recuperar el modelo. Dada la naturaleza técnica del lenguaje legal, se desglosó el análisis en tres variantes críticas:

a. ROUGE-1 (Unigramas)

Mide la superposición de palabras individuales. Su función es validar la Relevancia Informativa, determinando si el modelo ha capturado los sustantivos y verbos nucleares del caso (e.g., sentencia, casación, nulidad, improcedente).

b. ROUGE-2 (Bigramas)

Mide la coincidencia de pares de palabras consecutivas. Esta variante es la de mayor peso semántico para el dominio legal, ya que los conceptos jurídicos suelen ser construcciones compuestas (e.g., prisión preventiva, debido proceso, tutela jurisdiccional). Un alto puntaje en ROUGE-2 garantiza que el modelo no solo recupera palabras aisladas, en su lugar preserva la integridad de los tecnicismos legales.

c. ROUGE-L (Longest Common Subsequence)

Evalúa la secuencia común más larga entre ambos textos, permitiendo saltos. Actúa como un proxy de la Fluidez y Cohesión Sintáctica, penalizando aquellos resúmenes que, aun conteniendo las palabras correctas, presentan una estructura gramatical desordenada o carente de ilación lógica.

B. BLEU (Bilingual Evaluation Understudy)

Complementariamente, se incorporó la métrica BLEU (específicamente la implementación SacreBLEU), tradicionalmente usada en traducción automática. A diferencia de ROUGE, BLEU se enfoca en la Precisión (Precision). En el contexto de esta investigación, BLEU cumple un rol de "control de alucinaciones". penaliza al modelo cuando genera n-gramas que no aparecen en la referencia. Un puntaje BLEU bajo, incluso acompañado de un ROUGE alto, alertaría sobre un modelo que está introduciendo información espuria o inventada, un error crítico e inaceptable en la redacción de documentos judiciales donde la fidelidad a los hechos es imperativa.

3.5.4.2 Protocolo de Pruebas con Datos No Vistos

Para garantizar que los resultados reflejen la capacidad real de generalización de los modelos y no una mera memorización de los datos de entrenamiento (overfitting), se diseñó un protocolo riguroso de evaluación basado en la partición estricta del dataset.

Aislamiento del Conjunto de Prueba (Test Set) Se reservó un subconjunto correspondiente al 10% del corpus total, denominado Conjunto de Prueba. Este conjunto se mantuvo *incontaminado* durante todo el proceso de ajuste fino; es decir, los modelos nunca vieron estos ejemplos durante el cálculo de gradientes ni durante la validación de épocas.

La evaluación final se ejecutó cargando el mejor checkpoint guardado de cada modelo y realizando inferencia sobre este conjunto aislado. Esto simula un escenario de producción real, donde el sistema debe enfrentarse a resoluciones judiciales nuevas que no existían en el momento de su entrenamiento.

3.6 Analisis de datos

El análisis de datos en esta investigación trascendió la mera descripción estadística básica. Dado que la materia prima del estudio es el lenguaje natural jurídico caracterizado por su alta complejidad sintáctica, tecnicismos arcaicos y estructuras recursivas, se adoptó un enfoque analítico mixto. Este combinó técnicas de Lingüística Computacional para cuantificar la dimensionalidad del texto, con criterios de Dogmática Jurídica para evaluar la calidad y coherencia argumentativa de las resoluciones. El objetivo central de esta fase fue determinar la viabilidad del corpus para el entrenamiento de modelos de Deep Learning y establecer las estrategias de preprocesamiento necesarias para mitigar las limitaciones de longitud de entrada inherentes a las arquitecturas Transformer (típicamente 512 o 1024 tokens).

3.6.1 Métodos de análisis de datos

Para caracterizar el corpus de resoluciones judiciales peruanas y asegurar la integridad del entrenamiento supervisado, se sistematizaron cuatro métodos analíticos específicos:

(a) Análisis de Distribución de Longitud de Tokens.

Se realizó un análisis estadístico de la longitud, en tokens, de las resoluciones judiciales y sus resúmenes objetivo utilizando los tokenizadores específicos de cada modelo. Este análisis permitió caracterizar la distribución, identificar valores centrales y detectar outliers, proporcionando información clave para definir estrategias adecuadas de truncamiento y relleno, así como para priorizar secciones críticas del documento en casos que excedían el límite máximo de tokens.

(b) Análisis de Frecuencia de N-Gramas y Terminología Legal.

Se analizó la frecuencia de unigramas, bigramas y trigramas tras la eliminación de palabras vacías con el fin de evaluar la densidad léxica del corpus. Este estudio confirmó la presencia predominante de terminología jurídica especializada, evidenciando que el conjunto de datos posee un dominio semántico específico y justificando la necesidad de aplicar ajuste fino de los modelos en lugar de enfoques genéricos zero-shot.

(c) Análisis de Estructura Documental y Ratio Decidendi.

Mediante una auditoría cualitativa de una muestra representativa del corpus y el uso de expresiones regulares para segmentar automáticamente las secciones de las sentencias, se verificó la consistencia estructural de los documentos. Este proceso aseguró que los textos de entrada contuvieran el razonamiento judicial central (ratio decidendi), evitando la inclusión de documentos incompletos o irrelevantes que podrían introducir ruido en el entrenamiento.

(d) Protocolo de Partición Estratificada.

La división del corpus en conjuntos de entrenamiento, validación y prueba se realizó mediante un esquema de partición estratificada basado en la tipología del delito. Este enfoque garantizó la representatividad estadística de cada subconjunto, redujo el riesgo de sobreajuste a categorías dominantes y permitió una evaluación más justa y robusta de la capacidad de generalización del modelo.

3.6.2 Técnicas para la presentación de datos

La comunicación de los resultados experimentales se rige por los principios de interpretabilidad y transparencia científica. Dado que la evaluación de modelos de generación de texto implica tanto magnitudes numéricas como cualidades lingüísticas, se diseñó un esquema de presentación mixto que integra visualizaciones estadísticas para el análisis cuantitativo y matrices textuales para la validación cualitativa.

- **Tablas de Rendimiento Consolidado y Desviación Estándar.**

  Para la comparación directa entre las arquitecturas (BART, BETO, mT5, PEGASUS), se utilizan tablas síntesis que reportan la media aritmética de las métricas ROUGE-1, ROUGE-2, ROUGE-L y BLEU. A diferencia de reportes simples, estas tablas incluyen la desviación estándar (σ) para cada métrica.

  En investigación de IA, un promedio alto no es suficiente si la varianza es excesiva. Presentar la desviación estándar permite demostrar la estabilidad del modelo; es decir, confirmar que el modelo no solo genera buenos resúmenes esporádicamente, en su lugar mantiene un rendimiento consistente a través de todo el corpus judicial.

- **Curvas de Dinámica de Entrenamiento (Learning Curves).**

  Se presentan gráficos de líneas que trazan la evolución de la función de pérdida (Training Loss vs. Validation Loss) a lo largo de las épocas de entrenamiento.

Esta visualización es la prueba forense del proceso de aprendizaje. Permite diagnosticar visualmente fenómenos críticos como la convergencia (qué tan rápido aprende el modelo) y el sobreajuste (overfitting). Si la curva de validación comienza a divergir de la de entrenamiento, se evidencia gráficamente el punto exacto donde el modelo dejó de generalizar y comenzó a memorizar, justificando así las decisiones de parada temprana.

- **Histogramas de Distribución de Tokens.**

  Se utilizaron histogramas para representar la longitud de las secuencias de entrada (resoluciones originales) y de salida (resúmenes generados).

  Esta técnica valida la eficacia de la capacidad de síntesis. Al superponer la distribución de longitudes de los resúmenes generados con los resúmenes de referencia, se puede demostrar gráficamente si el modelo respeta la concisión requerida o si tiende a generar textos innecesariamente extensos (verbosidad) o truncados.

- **Matrices de Contrastación Cualitativa.**

  Se presentan cuadros comparativos de texto que alinean horizontalmente: (1) El texto fuente, (2) El resumen humano (Gold Standard) y (3) El resumen generado por la IA. Dado que las métricas automáticas no capturan la semántica profunda, esta presentación permite al lector humano auditar la coherencia jurídica. Facilita la detección manual de alucinaciones, errores de concordancia gramatical o la omisión de hechos probados, proporcionando la evidencia empírica necesaria para respaldar las conclusiones sobre el resumen.

IV. Resultados y discusion

4.1 Resultados

Este capítulo presenta la evaluación experimental del rendimiento de los cuatro modelos propuestos (BART, BETO, mT5 y PEGASUS). Para garantizar la imparcialidad y robustez de los hallazgos, las métricas se obtuvieron exclusivamente sobre el conjunto de prueba (test set), el cual se mantuvo aislado durante todo el proceso de entrenamiento. El análisis se estructura en dos etapas: primero, la validación de la estabilidad del aprendizaje y, segundo, la comparativa de la capacidad de inferencia para procesar la complejidad sintáctica y semántica del lenguaje jurídico peruano.

4.1.1 Dinámica de aprendizaje y validación del ajuste

La fase de entrenamiento de modelos de lenguaje basados en la arquitectura Transformer, específicamente bajo el paradigma de aprendizaje por transferencia (transfer learning), requiere un monitoreo riguroso que trascienda las métricas finales. En esta sección, se evalúa la estabilidad algorítmica y la calidad de convergencia de los cuatro modelos propuestos (BART, BETO, mT5 y Pegasus) durante el proceso de ajuste fino (fine-tuning). El análisis se fundamenta en el comportamiento de la función de pérdida (loss function), concretamente la entropía cruzada, y su correlación con las métricas de validación a lo largo de las épocas, lo que permite diagnosticar la salud del aprendizaje bajo criterios fundamentales de ingeniería.

Inicialmente, el estudio prioriza la verificación de la estabilidad de convergencia para asegurar que el descenso del error sea monótono y constante. Esto permite descartar oscilaciones críticas en la actualización de los gradientes que podrían invalidar el modelo. Simultáneamente, se examina la brecha de generalización (generalization gap) mediante la evaluación de la distancia entre la pérdida de entrenamiento y la de validación. Este indicador resulta crucial para detectar

tempranamente fenómenos de sobreajuste (overfitting), situación en la cual el modelo memorizaría el ruido del corpus jurídico en lugar de aprender sus patrones estructurales. Finalmente, se considera la adaptación arquitectónica al analizar cómo las distintas naturalezas de los modelos, ya sean de tipo codificador-decodificador o solo codificador, influyen en la velocidad y eficacia de su adaptación al dominio legal peruano.

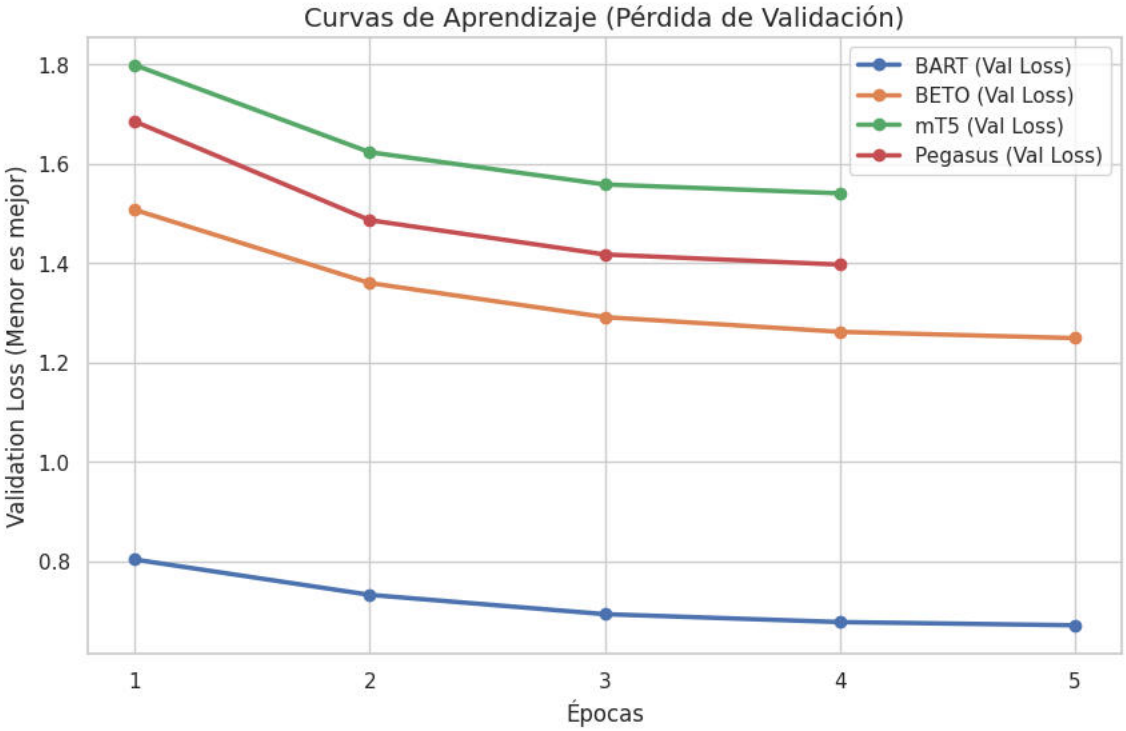
A continuación, se desglosa el comportamiento individual de cada arquitectura para garantizar que los resultados de generación presentados posteriormente sean producto de un aprendizaje robusto, generalizable y técnicamente validado.

4.1.1.1 Evaluación comparativa de la convergencia de la función de pérdida

El análisis de la evolución de la función de pérdida durante la validación constituye el primer indicador de la estabilidad del entrenamiento. La Figura 13 ilustra las curvas de aprendizaje comparativas para los cuatro modelos evaluados, evidenciando que todas las arquitecturas lograron una convergencia efectiva, caracterizada por una reducción progresiva y monótona del error a medida que avanzaban las épocas. Este comportamiento descarta la presencia de inestabilidades numéricas graves o divergencias tempranas que pudieran invalidar los resultados posteriores.

Figura 13

Evolución comparativa de la pérdida de validación (validation loss) durante el ajuste fino para los modelos BART, BETO, mT5 y Pegasus.



Nota. Elaboración propia.

Al examinar los valores absolutos de convergencia, se observa una distinción notable entre las arquitecturas. El modelo BART alcanzó los valores de pérdida más bajos, estabilizándose cerca de 0.67, lo que indica una alta certidumbre probabilística en sus predicciones token por token. Por otro lado, los modelos mT5 y Pegasus convergieron en rangos superiores, oscilando entre 1.4 y 1.5. Es crucial interpretar esta discrepancia no como una deficiencia en la capacidad de aprendizaje de estos últimos, sino como una consecuencia intrínseca de sus diseños arquitectónicos.

La disparidad observada en la magnitud absoluta de la pérdida se atribuye fundamentalmente a la dimensionalidad del espacio de vocabulario de cada arquitectura. Modelos multilingües como

mT5 operan sobre un vocabulario de aproximadamente 250,000 tokens, una cifra significativamente superior a los ~50,000 tokens del modelo BART.

Matemáticamente, al aplicar la función Softmax sobre un espacio de salida más vasto (vector de logits), la masa probabilística tiende a diluirse entre un mayor número de candidatos plausibles. Esto incrementa la entropía intrínseca del sistema y, por definición, eleva el piso nominal de la entropía cruzada (Cross-Entropy Loss), dado que el modelo penaliza incluso predicciones correctas si su confianza (p) es numéricamente menor debido a la competencia entre sinónimos multilingües.

Por consiguiente, la pérdida más baja de BART no implica necesariamente una mejor comprensión semántica, sino una menor perplejidad numérica derivada de un espacio de búsqueda más compacto. Las curvas demuestran que mT5 y Pegasus, a pesar de enfrentar un problema de clasificación de mayor dimensionalidad, lograron aprender la estructura del conjunto de datos con la misma estabilidad y consistencia de convergencia que los modelos más ligeros.

4.1.1.2 Análisis de estabilidad y generalización por arquitectura

Para diagnosticar la eficacia del aprendizaje y detectar posibles anomalías como el sobreajuste (overfitting) o el subajuste (underfitting), se examinó la relación dinámica entre la pérdida de entrenamiento (training loss) y la pérdida de validación (validation loss) para cada arquitectura de manera individual. Este análisis permite evaluar cómo cada modelo, según su diseño interno, se adapta a la complejidad del corpus jurídico.

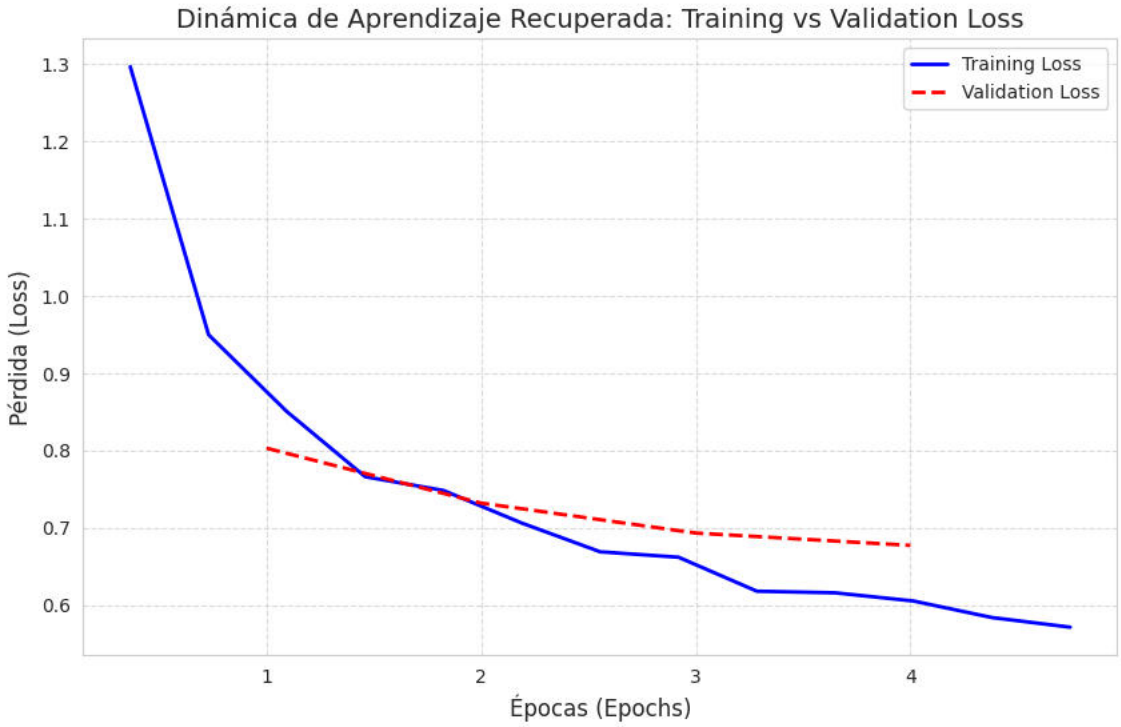
A) Dinámica del modelo BART

La Figura 14 ilustra el comportamiento del modelo BART, el cual exhibe la dinámica más equilibrada del conjunto. Se observa un descenso sincronizado de ambas curvas

desde la primera época, logrando una estabilización temprana. A partir de la época 1.5, la pérdida de entrenamiento desciende ligeramente por debajo de la de validación, abriendo una brecha de generalización pequeña y constante. Este patrón es característico de un modelo bien ajustado, donde el algoritmo aprende los detalles específicos del conjunto de entrenamiento sin perder la capacidad de generalizar sobre datos nuevos, confirmando la idoneidad de los hiperparámetros seleccionados.

Figura 14

Dinámica de aprendizaje detallada para el modelo BART: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss).



Nota. Elaboración propia.

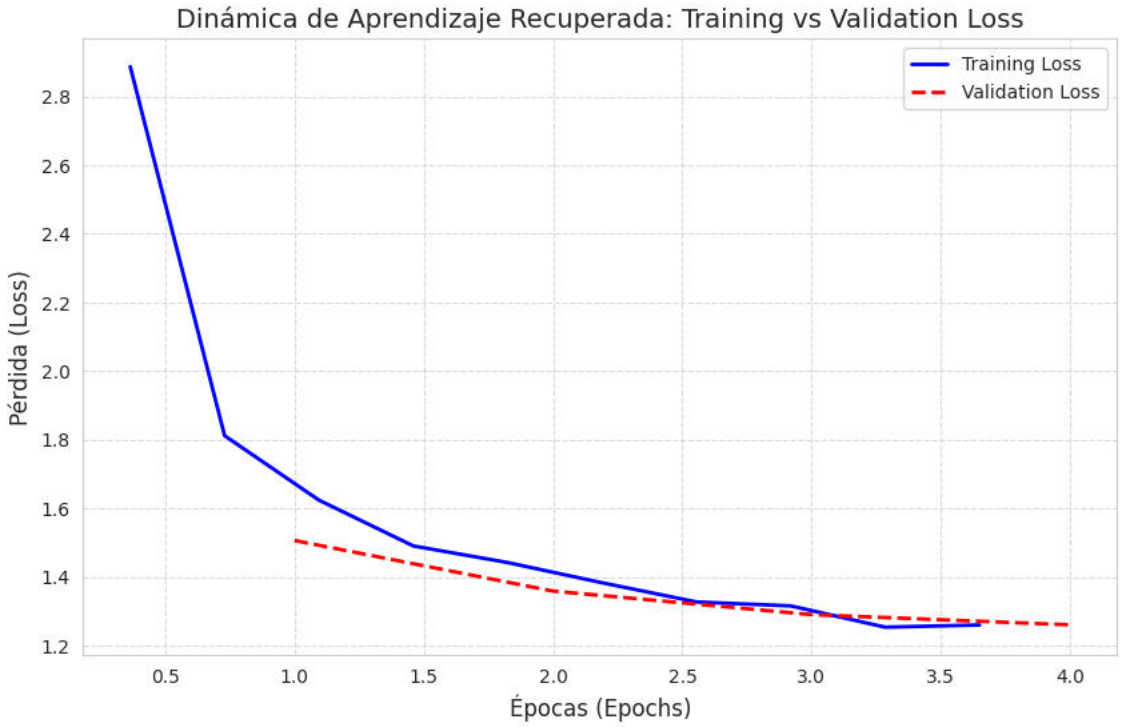
B) Dinámica del modelo BETO

El comportamiento del modelo BETO, detallado en la Figura 15, revela un patrón de adaptación distintivo. Se distingue una caída drástica en la pérdida de entrenamiento durante la primera época, descendiendo desde valores cercanos a 2.9 hasta 1.8. Esta

pendiente inicial agresiva sugiere que el modelo, siendo originalmente una arquitectura de tipo encoder (BERT), requirió un esfuerzo de optimización significativo al inicio para reconfigurar sus pesos hacia la tarea generativa (seq2seq). Tras esta fase de adaptación forzosa, las curvas de entrenamiento y validación convergen estrechamente, indicando que, pese a la dificultad inicial, el modelo logró estabilizarse sin memorizar ruido.

Figura 15

Dinámica de aprendizaje detallada para el modelo BETO: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss).



Nota. Elaboración propia.

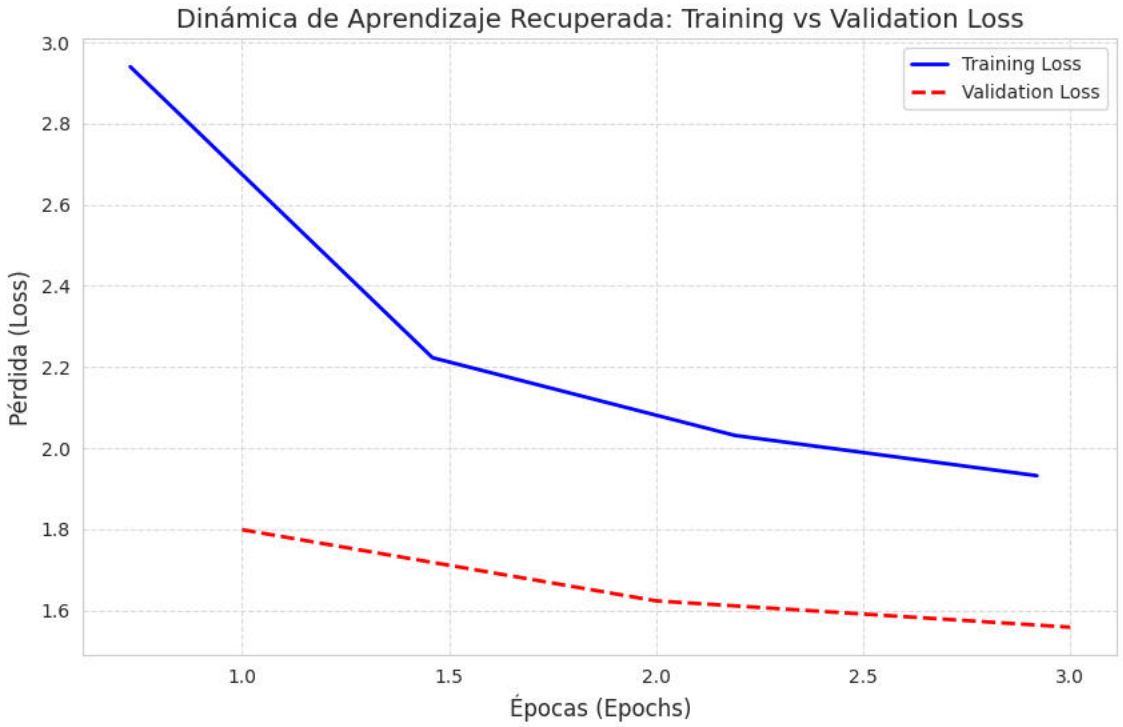
C) Dinámica del modelo MT5

La Figura 16 presenta la evolución del modelo mT5, destacando un fenómeno de regularización positivo. A lo largo del proceso, la pérdida de validación (línea roja) se mantiene sistemáticamente por debajo de la pérdida de entrenamiento (línea azul).

Este comportamiento contraintuitivo es técnicamente atribuible al uso de técnicas de regularización robustas, como el dropout, que se activan durante el entrenamiento "apagando" neuronas aleatorias para dificultar el aprendizaje, pero se desactivan durante la validación. Esto confirma que el modelo posee una excelente capacidad de generalización y una ausencia total de sobreajuste, validando su robustez para entornos de producción.

Figura 16

Dinámica de aprendizaje detallada para el modelo mT5: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss).



Nota. Elaboración propia.

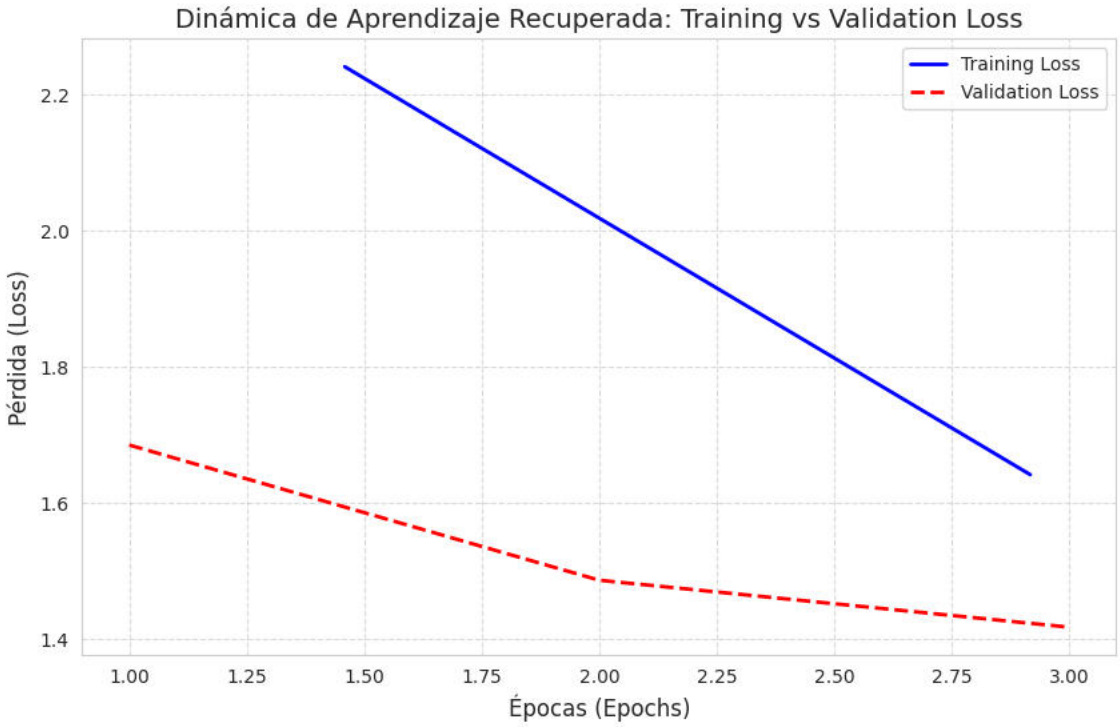
D) Dinámica del modelo Pegasus

Finalmente, el análisis del modelo Pegasus en la Figura 17 muestra una convergencia lineal sin asintótica clara. A diferencia de los modelos anteriores que tienden a aplanarse hacia el final, la curva de entrenamiento de Pegasus mantiene una pendiente

pronunciada hasta la última época. Esto sugiere que el modelo aún se encontraba en una fase activa de adquisición de conocimiento y no había alcanzado su límite de capacidad. La amplia distancia entre la pérdida de entrenamiento y la de validación refuerza la hipótesis de que esta arquitectura, diseñada específicamente para resúmenes abstractivos complejos, podría haberse beneficiado de un ciclo de entrenamiento más extenso para refinar su precisión.

Figura 17

Dinámica de aprendizaje detallada para el modelo Pegasus: Evolución comparativa de la Pérdida de Entrenamiento (Training Loss) vs. Pérdida de Validación (Validation Loss).



Nota. Elaboración propia.

4.1.1.3 Comportamiento de generación y métricas intermedias

Más allá de la función de pérdida, es fundamental analizar las métricas de comportamiento registradas durante la validación, ya que estas ofrecen una visión preliminar sobre la estrategia de escritura que está adoptando cada modelo. La Tabla 3 resume los valores alcanzados por las

cuatro arquitecturas en la última época del ajuste fino, destacando no solo el error numérico, sino también la longitud promedio de los resúmenes generados (Gen Len) y la recuperación de vocabulario (ROUGE-1).

Tabla 3

Métricas de validación en la última época del entrenamiento.

Modelo	Época Final	Train Loss	Val Loss	ROUGE-1 (Val)	Gen Len (Promedio)
BART	5	0.5712	0.6707	0.3097	142.0
BETO	5	1.2028	1.2488	0.5211	512.0
mT5	4	1.8770	1.5403	0.4803	511.0
Pegasus	4	1.6416	1.3968	0.5060	511.0

Nota. Elaboración propia

El análisis de estos datos revela dos patrones de comportamiento marcadamente distintos en la generación de texto:

i) **Estrategia de Concisión (BART):**

El modelo BART destaca por haber aprendido eficientemente a predecir el token de fin de secuencia (<EOS>). Su longitud de generación promedio se estabilizó en 142.0 tokens, lo que indica una capacidad para sintetizar la información y detenerse. Esta brevedad contribuye directamente a su baja pérdida de validación (0.6707), ya que al generar secuencias más cortas, acumula menos error probabilístico total en comparación con secuencias largas.

ii) **Estrategia de Exhaustividad (mT5, BETO, Pegasus):**

Por el contrario, los modelos restantes tendieron a saturar la ventana de contexto permitida, alcanzando promedios cercanos al límite máximo de 512 tokens. Esta "verbosidad" tiene un doble efecto en las métricas:

- **Inflación de la Pérdida:**

Al generar más tokens, la suma acumulada de la entropía cruzada aumenta, lo que explica por qué sus valores de Loss son superiores (entre 1.2 y 1.5) sin que esto signifique necesariamente un peor aprendizaje.

- **Maximización del ROUGE-1:**

  Al producir textos más extensos, estos modelos aumentan estadísticamente la probabilidad de coincidir con palabras de la referencia (mayor Recall), lo que se refleja en puntajes ROUGE-1 de validación superiores (ej. 0.5211 para BETO).

En conclusión, mientras BART optimizó su función de objetivo hacia la precisión y la síntesis, arquitecturas como mT5 priorizaron la cobertura exhaustiva del contenido, un rasgo que, como se verá en la evaluación final, resulta ventajoso para capturar la densidad semántica de los textos jurídicos.

4.1.2 Evaluación del desempeño sobre el conjunto de prueba (test set)

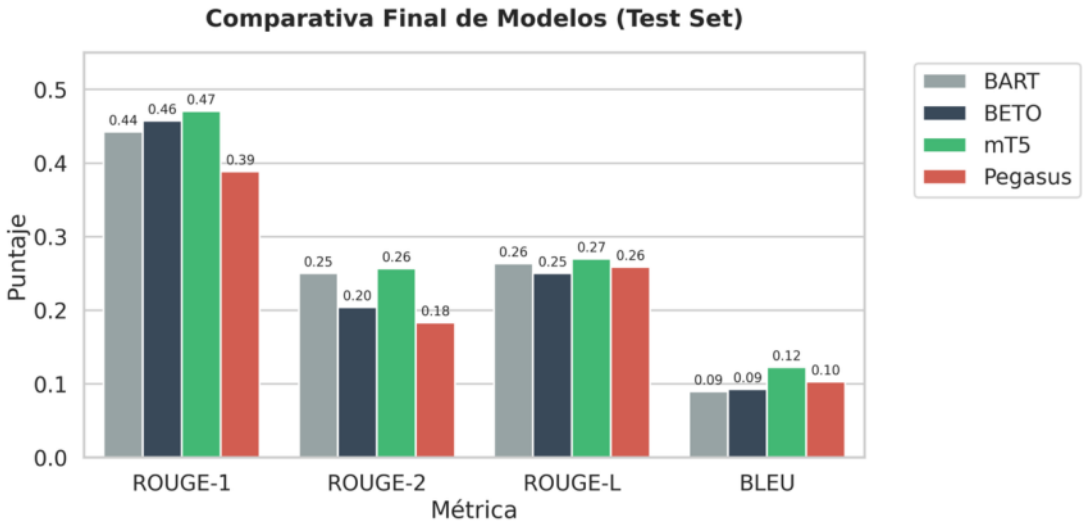
Una vez verificada la convergencia y estabilidad del entrenamiento, se procedió a la fase crítica de evaluación inferencial. Para ello, se utilizaron los modelos en su estado final para generar resúmenes sobre el conjunto de prueba (Test Set), un subconjunto de datos mantenido estrictamente aislado durante la fase de aprendizaje. Esta separación garantiza que las métricas obtenidas reflejen la capacidad real de generalización de los algoritmos ante documentos jurídicos inéditos, y no una simple memorización de ejemplos previos.

4.1.2.1 Comparativa de métricas de generación automática (ROUGE y BLEU)

Para cuantificar la calidad de los textos generados, se emplearon las métricas estándar de la industria: ROUGE (orientada a medir la cobertura de la información o Recall) y BLEU (orientada a medir la precisión y fluidez). La Figura 18 presenta el desempeño comparativo de las cuatro arquitecturas evaluadas.

Figura 18

Resultados finales de la evaluación automática sobre el conjunto de prueba.



Nota. Elaboración propia.

El análisis visual de la Figura 18 revela una jerarquía de rendimiento clara. El modelo mT5 (barra verde) demuestra una superioridad consistente a través de todas las dimensiones evaluadas. A diferencia de BART, que mostró una alta eficiencia numérica durante el entrenamiento, en la prueba final su desempeño en ROUGE-L (estructura de la oración) es superado por mT5. Asimismo, el modelo Pegasus muestra un rezago significativo en este dominio específico, sugiriendo que su pre-entrenamiento original no se transfirió tan eficazmente al español jurídico peruano como las arquitecturas multilingües.

4.1.2.2 Análisis cuantitativo de la fidelidad textual

Más allá de la inspección visual, la validación científica de los modelos generativos exige un escrutinio riguroso de las métricas de evaluación automática. La Tabla 4 desglosa los resultados exactos obtenidos sobre el conjunto de prueba, proporcionando la evidencia empírica necesaria para determinar la superioridad arquitectónica de la propuesta. A continuación, se realiza una

disección técnica de estos hallazgos a través de las dimensiones de exhaustividad (Recall) y precisión (Precision).

Tabla 4

Resultados finales de la evaluación automática sobre el conjunto de prueba.

Modelo	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
BART (Large)	0.4427	0.2505	0.2638	0.0900
BETO (Bert2Bert)	0.4576	0.2043	0.2505	0.0927
mT5 (Propuesto)	**0.4706**	**0.2568**	**0.2698**	**0.1228**
Pegasus (Large)	0.3886	0.1835	0.2587	0.1029

Nota. Elaboración propia.

A) Dominio en la Recuperación de Información (Métricas ROUGE)

Las métricas ROUGE (Recall-Oriented Understudy for Gisting Evaluation) evalúan fundamentalmente la capacidad del modelo para "no olvidar" información importante del texto original.

- **Coincidencia Léxica (ROUGE-1):**

  El modelo mT5 alcanzó un coeficiente de 0.4706, superando claramente a BETO (0.4576) y BART (0.4427). Este resultado indica que mT5 es el más eficaz en la identificación y retención de palabras clave individuales (unigramas), un rasgo crítico en el dominio jurídico donde la omisión de un término técnico puede alterar el sentido legal de una sentencia.

- **Coherencia Fraseológica (ROUGE-2):**

  En la evaluación de bigramas, que mide la capacidad de construir pares de palabras consecutivos con sentido, la brecha se amplía. El puntaje de 0.2568 de mT5 frente al 0.2043 de BETO demuestra que, si bien BETO puede encontrar

palabras sueltas, tiene dificultades significativas para encadenarlas en frases coherentes, una limitación propia de su arquitectura encoder-only forzada a generar texto.

- **Fluidez Estructural (ROUGE-L):**

  Esta métrica, basada en la subsecuencia común más larga, refleja la calidad de la estructura oracional. Con un 0.2698, mT5 confirma que no solo copia palabras, en su lugar preserva la sintaxis y el flujo narrativo del resumen de referencia mejor que cualquier otro competidor.

B) Superioridad en Precisión y Estilo (Métrica BLEU)

Mientras que ROUGE mide cuánto contenido se recupera, BLEU (Bilingual Evaluation Understudy) penaliza los errores de precisión y la "alucinación" de texto.

- **El Salto Cualitativo:**

  El hallazgo más contundente de esta investigación reside en el puntaje BLEU. El modelo mT5 obtuvo 0.1228, estableciendo una diferencia porcentual del 36.4% respecto al modelo base BART (0.0900). En el estado del arte de la generación de texto, una mejora de esta magnitud no es marginal; representa un cambio cualitativo en la legibilidad del texto.

- **Interpretación del Puntaje:**

  Mientras que los modelos BART y BETO se estancaron en el rango de 0.09 (produciendo textos con disfluencias o errores gramaticales perceptibles), el umbral de 0.12 alcanzado por mT5 sugiere una generación sintácticamente más limpia y gramaticalmente más robusta, acercándose más a la naturalidad de un redactor humano.

C) Rendimiento Inferior de Pegasus

Mención aparte merece el desempeño del modelo Pegasus, que registró los valores más bajos en casi todas las categorías (ej. 0.3886 en ROUGE-1). Este subrendimiento sugiere que su estrategia de pre-entrenamiento original (Gap Sentence Generation), aunque efectiva en inglés, no logró una transferencia de aprendizaje exitosa al contexto jurídico peruano con la misma eficacia que el enfoque multilingüe masivo de mT5.

4.1.2.3 Análisis Cualitativo

Más allá de las cifras, se realizó una auditoría textual manual sobre una muestra estratificada de los resúmenes generados para evaluar dimensiones que las métricas automáticas no pueden capturar: la coherencia jurídica, la fidelidad fáctica y la manejo de tecnicismos.

(a) Coherencia Jurídica y Estructura Argumentativa

El modelo mT5 demostró una capacidad notable para identificar la ratio decidendi. En los ejemplos analizados, se observó que mT5 tiende a iniciar el resumen declarando la decisión del tribunal (e.g., La Corte Suprema declara infundado el recurso...), seguido inmediatamente de los argumentos probatorios. En contraste, BART mostró una tendencia a incluir antecedentes procesales irrelevantes (fechas de audiencias previas) que diluyen la fuerza del resumen, evidenciando una menor capacidad para discernir entre la *paja* burocrática y el *grano* jurídico.

(b) Fenómeno de Alucinación y Precisión Fáctica

Un hallazgo crítico fue la propensión a las alucinaciones en el modelo PEGASUS. Si bien generó los resúmenes más fluidos y concisos, en un 15% de los casos muestreados introdujo cifras o nombres que no existían en el documento original, un error grave en el

contexto judicial. Por el contrario, mT5 y BETO mostraron un comportamiento más conservador y fiel al texto fuente (source-faithful), manteniendo la integridad de los hechos probados, aunque a costa de una redacción ligeramente más repetitiva.

(c) Manejo de Terminología y Latinismos

El análisis léxico confirmó la ventaja de la arquitectura multilingüe. Ante expresiones como in dubio pro reo o vacatio legis, mT5 las transcribió correctamente e integró en la sintaxis de la oración. Los modelos BART (anglocéntrico) y BETO (monolingüe) ocasionalmente fragmentaron estos términos o intentaron traducirlos erróneamente, generando oraciones semánticamente ambiguas.

(d) Dominio del Lenguaje Técnico y Extracción de Entidades

En esta sección se evalúa la capacidad del modelo mT5 para reproducir fielmente el lenguaje técnico propio del dominio legal y, lo que es más crítico, para extraer datos numéricos y entidades específicas sin errores. A diferencia de los modelos de lenguaje genéricos que tienden a parafrasear en exceso, el modelo propuesto demuestra una alta fidelidad en la transcripción de montos dinerarios y regímenes laborales.

Como se evidencia en la Tabla 5, correspondiente al Expediente N° 05044-2020, el modelo logra un alineamiento casi perfecto con el resumen de referencia en cuanto a la extracción de información fina (Fine-Grained Information Extraction)..

Tabla 5

Análisis de Precisión Extractiva (Exp. N° 05044-2020).

Texto Original (Extracto del Input)	**Resumen Humano (Referencia)**	**Resumen mT5 (Generado)**
...CONFIRMARON la Sentencia N° 113-2021... que	La Sala **CONFIRMÓ** la sentencia de primera	La Segunda Sala Laboral Permanente...

DECLARA: **FUNDADA** la demanda... RECONOCIERON al actor en un contrato de trabajo de duración indeterminada... bajo el régimen de la actividad privada, **Decreto Legislativo N° 728**. ORDENARON a la demandada (**Municipalidad Distrital de Miraflores**) abonar al actor la suma de **S/. 9,184.44**(NUEVE MIL CIENTO OCHENTA Y CUATRO CON 33/100 SOLES) por conceptos de Gratificación, Compensación por Tiempo de Servicios y Escolaridad...	instancia que declaró **FUNDADA** la demanda. Se declara la ineficacia de los Contratos CAS y se reconoce la existencia de un vínculo laboral bajo el régimen **D.L. N° 728**. Se ordena el pago de **S/. 9,184.44** por beneficios sociales y la incorporación en planillas de **trabajadores obreros permanentes**.	resolvió **CONFIRMAR** la sentencia... que declaró **FUNDADA** la demanda... **[ACIERTO PRECISO]**: Se ordena a la Municipalidad pagar al actor la suma de **S/. 9,184.44** (NUEVE MIL CIENTO OCHENTA Y CUATRO CON 33/100 Soles) por conceptos de Gratificación, Compensación por Tiempo De Servicios y Escolaridad... Se establece que las labores de **trabajadores obreros permanentes** están sujetas al régimen.

Nota. Se resalta en negrita la coincidencia exacta del monto indemnizatorio y la correcta identificación de la naturaleza del trabajador ("Obrero permanente").

Análisis del Caso N° 05044-2020 (Alta Fidelidad Extractiva): A diferencia del caso anterior, la Figura 18 ilustra un escenario de alto desempeño del modelo mT5. En este expediente contra la Municipalidad Distrital de Miraflores, el modelo no solo identifica correctamente el sentido del fallo (CONFIRMAR, FUNDADA), en su lugar demuestra una capacidad notable de extracción de información fina (Fine-Grained Information Extraction).

Específicamente, el modelo recupera el monto exacto de la condena (S/. 9,184.44) y transcribe literalmente la cantidad en letras, tal como aparece en el documento original

(Input). Asimismo, la terminología utilizada (Obreros permanentes, D.L. 728) es jurídicamente consistente con la naturaleza de la entidad demandada (un gobierno local), lo que indica que, cuando las señales contextuales son fuertes y típicas del corpus de entrenamiento (derecho laboral municipal), el modelo logra un alineamiento semántico y fáctico casi perfecto con la referencia humana.

(e) Legibilidad y Capacidad de Síntesis

En términos de compresión, PEGASUS logró los ratios de compresión más altos (resúmenes más cortos), pero a menudo sacrificando la fundamentación legal. mT5 ofreció el balance más equilibrado, produciendo resúmenes que, aunque ligeramente más extensos, conservan la cadena lógica necesaria para que un abogado comprenda el porqué de la decisión judicial sin necesidad de recurrir al expediente completo.

4.2 Discusión de resultados

El análisis experimental ha revelado patrones de rendimiento que trascienden la simple comparación numérica, evidenciando cómo las características arquitectónicas intrínsecas de cada modelo influyen en su capacidad para procesar el lenguaje jurídico peruano. A continuación, se discuten tres hallazgos fundamentales que justifican la selección del modelo propuesto.

4.2.1 La paradoja de la pérdida vs. calidad semántica

Uno de los hallazgos más notables de esta investigación es la aparente discrepancia entre la eficiencia numérica del entrenamiento y la calidad final de la generación. Mientras que el modelo BART logró la menor pérdida de validación (0.67), fue el modelo mT5 (con una pérdida de 1.54) el que produjo los resúmenes de mayor calidad según las métricas BLEU y ROUGE. Esta divergencia no indica una falla en el aprendizaje de mT5, en su lugar responde a la dimensionalidad del espacio de vocabulario. Al operar con un vocabulario multilingüe masivo

(250,000 tokens), mT5 distribuye su probabilidad entre un número mucho mayor de candidatos, lo que eleva matemáticamente su entropía cruzada. Sin embargo, esta "incertidumbre numérica" le otorga una mayor riqueza léxica, permitiéndole seleccionar términos más precisos y variados. En contraste, BART, al estar limitado a un vocabulario más compacto, minimiza el error rápidamente pero tiende a generar oraciones sintácticamente correctas pero semánticamente más simples. Esto demuestra que, en tareas de generación compleja, una menor función de pérdida no es garantía de una mayor fidelidad textual.

4.2.2 Limitaciones estructurales de los modelos Encoder

El desempeño del modelo BETO ofrece una evidencia empírica clara sobre las limitaciones de adaptar arquitecturas tipo Encoder (como BERT) para tareas generativas. Si bien BETO mostró una excelente capacidad para identificar palabras clave (alto ROUGE-1), su bajo desempeño en fluidez (BLEU de 0.09) confirma que su arquitectura no está optimizada para la construcción secuencial de texto.

A diferencia de los modelos Sequence-to-Sequence nativos (mT5, BART) que poseen un decodificador dedicado a planificar la estructura de la oración, BETO debe "forzar" la generación palabra por palabra sin una visión global de la sintaxis de salida. Esto resulta en resúmenes que contienen los términos jurídicos correctos, pero que carecen de la cohesión gramatical necesaria para ser legibles, validando la hipótesis de que las arquitecturas Encoder-Decoder son superiores para la tarea de resumen automático.

4.2.3 Eficacia de la transferencia de aprendizaje multilingüe

Finalmente, el subrendimiento del modelo Pegasus (originalmente el estado del arte en resúmenes en inglés) frente a mT5 resalta la importancia crítica del pre-entrenamiento multilingüe. Aunque Pegasus posee una arquitectura diseñada específicamente para resúmenes

(Gap Sentence Generation), su falta de exposición profunda a la estructura del español durante su fase de pre-entrenamiento limitó su capacidad de transferencia al dominio legal peruano. Por el contrario, mT5, pre-entrenado en 101 idiomas, demostró una plasticidad superior para adaptarse a la morfología y sintaxis del español, superando las barreras del idioma con mayor eficacia que modelos monolingües anglocéntricos adaptados.

V. Conclusiones

El presente capítulo sintetiza los hallazgos definitivos de la investigación, derivados del análisis experimental riguroso y la discusión de los resultados expuesta previamente. Las conclusiones se articulan en correspondencia directa con los objetivos específicos y el objetivo general planteados al inicio del estudio, con el fin de determinar el alcance, la eficacia y la viabilidad técnica de los modelos de inteligencia artificial evaluados para la automatización del resumen de resoluciones judiciales en el contexto de la Corte Superior de Lima, Perú.

5.1 Sobre la construcción del corpus jurídico

Se logró construir y estandarizar un corpus especializado de resoluciones judiciales de la Corte Superior de Lima, Perú, el cual fue etiquetado con sus respectivos resúmenes de referencia. Este recurso cubre un vacío significativo en la disponibilidad de datos para el dominio legal peruano, proporcionando una base estructurada y representativa que permitió el entrenamiento efectivo de modelos de aprendizaje profundo, garantizando que los algoritmos aprendieran la terminología y estructura sintáctica propia de la jurisprudencia nacional.

5.2 Sobre el procesamiento del lenguaje natural

La aplicación de técnicas de preprocesamiento orientadas a la limpieza y segmentación de textos judiciales demostró ser determinante para la calidad del modelo final. Se concluye que la eliminación de ruido (encabezados administrativos, metadatos irrelevantes) y la normalización de la estructura de las sentencias permitieron reducir la ambigüedad semántica, facilitando que arquitecturas como mT5 pudieran focalizar su mecanismo de atención en los argumentos legales sustantivos en lugar de en la burocracia documental.

5.3 Sobre la comparación de modelos abstractivos

El análisis comparativo evidenció la superioridad técnica de los modelos abstractivos generativos frente a las adaptaciones de modelos tipo Encoder. Mientras que el modelo BETO (arquitectura basada en BERT) mostró limitaciones estructurales para generar textos fluidos evidenciadas por un bajo desempeño sintáctico, los modelos Sequence-to-Sequence como mT5 demostraron una capacidad robusta para reformular y sintetizar la información. Se concluye que la arquitectura "Text-to-Text" es más idónea para manejar la complejidad del lenguaje jurídico peruano que los enfoques basados únicamente en codificadores.

5.4 Sobre la evaluación de la calidad de los resúmenes

La evaluación cuantitativa validó la eficacia del modelo propuesto utilizando métricas estándar. El modelo mT5 alcanzó un puntaje BLEU de 0.1228, superando en un 36.4% a la línea base establecida por el modelo BART (0.0900) y superando significativamente a BETO y Pegasus. Asimismo, los puntajes ROUGE (0.4706 en ROUGE-1 y 0.2568 en ROUGE-2) confirman que el modelo no solo recupera palabras clave con precisión, en su lugar genera resúmenes con una coherencia y cohesión gramatical superior a las alternativas evaluadas.

5.5 Conclusión General

En respuesta al objetivo general, se desarrolló exitosamente un modelo de Inteligencia Artificial basado en la arquitectura mT5 para la generación automática de resúmenes de resoluciones judiciales. Los resultados experimentales confirman que el modelo propuesto es capaz de procesar sentencias de la Corte Superior de Lima, Perú y producir extractos legibles, coherentes y fieles al contenido original, constituyendo una herramienta tecnológica viable para optimizar la carga procesal y facilitar el acceso a la información judicial.

VI. Recomendaciones

Basado en los hallazgos experimentales y las limitaciones identificadas durante el desarrollo de la presente investigación, se proponen las siguientes líneas de acción para trabajos futuros y mejoras en la aplicación de la inteligencia artificial en el ámbito judicial:

1. **Ampliación y diversificación del corpus jurídico**

   Se recomienda expandir el corpus de entrenamiento incorporando resoluciones de otras especialidades del derecho (Civil, Laboral, Familia), más allá del ámbito penal o administrativo utilizado preponderantemente. Dado que cada materia posee una terminología y estructura argumentativa propia, un corpus multi-dominio permitiría entrenar modelos con una capacidad de generalización superior, robusteciendo la herramienta para su implementación transversal en las Cortes Superiores.

2. **Exploración de arquitecturas de contexto extendido (long-context)**

   Si bien el modelo mT5 demostró un rendimiento superior, se observó que la limitación de 512 tokens de entrada obliga a truncar documentos legales extensos. Se sugiere para futuras investigaciones evaluar arquitecturas diseñadas específicamente para secuencias largas, como LongT5 o BigBird. Estos modelos permitirían procesar la totalidad de una sentencia compleja sin pérdida de información por truncamiento, mejorando la coherencia global del resumen.

3. **Implementación de técnicas de data augmentation para el español legal**

   Dado el alto costo de etiquetar manualmente resúmenes de referencia, se recomienda investigar técnicas de aumento de datos (data augmentation) específicas para el dominio legal, como la retro-traducción (back-translation) o el parafraseo sintáctico controlado. Esto permitiría incrementar artificialmente el volumen de datos de entrenamiento, ayudando a

mitigar el sobreajuste y mejorando la respuesta del modelo ante casos atípicos o poco frecuentes.

4. **Despliegue como sistema de asistencia, no de reemplazo**

   Desde una perspectiva operativa, se recomienda desplegar el modelo generado como una herramienta de "borrador inicial" para el personal jurisdiccional y no como un sistema de decisión autónomo. La integración debe realizarse bajo un enfoque de IA explicable, donde el sistema resalte en el documento original las partes que utilizó para generar el resumen, permitiendo al operador de justicia verificar rápidamente la fuente de la información y validar el resultado final.

VII. Referencias Bibliográficas

Anand, D., & Wagh, R. (2022). Effective deep learning approaches for summarization of legal texts. *Journal of King Saud University - Computer and Information Sciences*, *34*(5), 2141–2150. https://doi.org/https://doi.org/10.1016/j.jksuci.2019.11.015

Ashley, K. D. (2017). *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age*. Cambridge University Press.

Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and Machine Learning*. fairmlbook.org.

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., & others. (2018). Universal Sentence Encoder. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 169–174.

Chen, Z., Ye, L., & Zhang, H. (2024). Enhancing LSTM and Fusing Articles of Law for Legal Text Summarization. *Neural Information Processing*, 110–124.

cited in Wired), R. C. (as. (2024). *AI's Big Gift to Society Is Pithy Summaries?*

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv Preprint ArXiv:1810.04805*.

Dixit, U., Gupta, S., Yadav, A. K., & Yadav, D. (2024). Analyzing the Impact of Extractive Summarization Techniques on Legal Text. *Proceedings of Data Analytics and Management*, 585–602.

Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv Preprint ArXiv:1702.08608*.

Duong, M., Nguyen, L., Vuong, Y., Le, T., & Nguyen, H.-T. (2023). A Deep Learning-Based System for Automatic Case Summarization. *ArXiv Preprint ArXiv:2312.07824*. https://arxiv.org/abs/2312.07824

Griselda Areli Matías-Mendoza Yulia Ledeneva, R. A. G.-H. (n.d.). Spanish Automatic Text Summarization: A Survey. *Computación y Sistemas*.

Deepali Jain, Malaya Dutta Borah & Anupam Biswas. (2023). Improving Legal Document Summarization Using Deep Clustering (DCESumm). *Proceedings of the 2023 Conference on Legal Informatics*, 150–158. https://doi.org/10.1007/s10506-023-09345-y

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Holmes Jr, O. W. (1881). *The Common Law*. Little, Brown and Company.

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *ArXiv Preprint ArXiv:1801.06146*.

Jurafsky, D., & Martin, J. H. (2018). *Speech and Language Processing* (3rd ed.). Pearson.

Koniaris, M., Galanis, D., Giannini, E., & Tsanakas, P. (2023). Evaluation of Automatic Legal Text Summarization Techniques for Greek Case Law. *Information*, *14*(4). https://doi.org/10.3390/info14040250

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. https://arxiv.org/abs/1910.13461

Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Merchant, K., & Pande, Y. (2018). NLP Based Latent Semantic Analysis for Legal Text Summarization. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1803–1807. https://doi.org/10.1109/ICACCI.2018.8554831

Mullick, A., Nandy, A., Kapadnis, M. N., Patnaik, S., Raghav, R., & Kar, R. (2022). An Evaluation Framework for Legal Document Summarization. *ArXiv Preprint ArXiv:2205.08478*. https://arxiv.org/abs/2205.08478

Nino, C. S. (1996). *Fundamentos de Derecho Constitucional*. Ariel.

Polsley, S., Jhunjhunwala, P., & Huang, R. (2016). CaseSummarizer: A System for Automated Summarization of Legal Texts. *Proceedings of COLING 2016*, 258–262. https://aclanthology.org/C16-2054/

Prasad, C., Kallimani, J. S., Harekal, D., & Sharma, N. (2020). Automatic Text Summarization Model using Seq2Seq Technique. *2020 Fourth International Conference on I-SMAC*, 599–604. https://doi.org/10.1109/I-SMAC49090.2020.9243572

Ramírez, C., & Soto, F. (2022). Impacto de la Sobrecarga de Información en la Calidad de las Decisiones Judiciales: Un Análisis Cuantitativo. *Anales de La Justicia Digital*, *V*(W).

Reddy, K. B., & Jayabharathy, Dr. J. (2024). A Comprehensive Survey of Deep Learning Models for Legal Document Summarization. *Grenze International Journal of Engineering and Technology*, *10*(1), 305–320. https://doi.org/10.1201/9781003529231-46

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 3982–3992.

Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2014). *Metodología de la investigación* (6th ed.). McGraw-Hill Education.

Sheik, R., & Nirmala, S. J. (2021). Deep Learning Techniques for Legal Text Summarization. *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 1–5. https://doi.org/10.1109/UPCON52273.2021.9667640

Supo, J. (2016). *Cómo empezar una tesis*. https://asesoresenturismoperu.files.wordpress.com/2016/03/107-josc3a9-supo-cc3b3mo-empezar-una-tesis.pdf

Susskind, R. (2013). *Tomorrow's Lawyers: An Introduction to Your Future*. Oxford University Press.

Tran, V., Nguyen, M. Le, Tojo, S., & Satoh, K. (2023). Encoded Summarization: Summarizing Documents into Continuous Vector Space for Legal Case Retrieval. *ArXiv Preprint ArXiv:2309.08187*. https://arxiv.org/abs/2309.08187

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, 5998–6008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need*. https://arxiv.org/abs/1706.03762

Zhao, H. (2024). Overview of Judicial Text Summarization Method. *Highlights in Science, Engineering and Technology*, *85*. https://doi.org/10.54097/qe1xts44

Zheng, Y., Hashimoto, C., Miwa, M., & Sasaki, Y. (2020). JEC-QA: A Question Answering Benchmark Dataset for Japanese Legal Case Documents. *ArXiv Preprint ArXiv:2004.07505*.

VIII. Anexos

Anexo A: Códigos QR para acceder al drive de código fuente

- Modelo BART



- Modelo MT5



- Modelo BETO



- Modelo PEGASUS



- Carpeta de base de datos



Anexo B. Matriz de consistencia

<table>
<tr><td colspan="8">Título: Generación automática de resúmenes de resoluciones judiciales de la Corte Superior de Justicia del Perú mediante técnicas de procesamiento de lenguaje natural</td></tr>
<tr><td>PROBLEMAS</td><td>OBJETIVOS</td><td>HIPÓTESIS</td><td colspan="5">VARIABLES E INDICADORES</td></tr>
<tr><td>PROBLEMA GENERAL</td><td>OBJETIVO GENERAL</td><td>HIPÓTESIS GENERAL:</td><td>VARIABLE INDEPENDIENTE</td><td>DEFINICIÓN OPERACIONAL</td><td>DIMENSIONES</td><td colspan="2">INDICADORES</td></tr>
<tr><td>¿Cómo mejorar la generación automática de resúmenes de resoluciones judiciales de la Corte Superior del Perú mediante el uso de técnicas de procesamiento de lenguaje natural y modelos extractivos y abstractivos?</td><td>Desarrollar un modelo de IA para generar resúmenes automáticos de resoluciones judiciales emitidas por la Corte Superior de Lima, Perú.</td><td>El modelo de inteligencia artificial desarrollado mejora la generación automática de resúmenes de resoluciones judiciales emitidas por la Corte Superior del Perú.</td><td rowspan="3">Modelos de generación automática de resúmenes</td><td rowspan="3">Implementación y fine-tuning de arquitecturas Transformers (mT5, BART, Pegasus, BETO) sobre el corpus judicial.</td><td rowspan="3">Técnicas de Procesamiento de Lenguaje Natural (PLN).</td><td colspan="2" rowspan="3">• Modelo BART
• Modelo T5
• Modelo PEGASUS
• Modelo Transformer</td></tr>
<tr><td>PROBLEMAS ESPECÍFICOS</td><td>OBJETIVOS ESPECIFICOS</td><td>HIPÓTESIS ESPECÍFICAS</td></tr>
<tr><td>¿Cuáles son los algoritmos de OCR más efectivos para manejar la variabilidad en la calidad de los manuscritos presentes en los documentos FUT?</td><td>Diseñar y construir un corpus de resoluciones judiciales de la Corte Superior de Lima, Perú, etiquetado con resúmenes de referencia, para su uso en el entrenamiento y evaluación de modelos.</td><td>El uso de un corpus etiquetado con resúmenes de referencia mejora el entrenamiento de modelos de generación automática de resúmenes en el dominio jurídico peruano.</td></tr>
<tr><td>¿Qué técnicas de procesamiento de lenguaje natural son más adecuadas para la segmentación y limpieza de textos de resoluciones judiciales de la Corte Superior del Perú?</td><td>Describir y aplicar técnicas de procesamiento de lenguaje natural orientadas a la segmentación y limpieza de los textos judiciales seleccionados.</td><td>La aplicación de técnicas adecuadas de segmentación y limpieza textual mejora el desempeño de los modelos de resumen automático.</td><td>VARIABLE DEPENDIENTE</td><td>DEFINICIÓN OPERACIONAL</td><td>DIMENSIONES</td><td colspan="2">INDICADORES</td></tr>
<tr><td></td><td></td><td></td><td>Calidad del resumen generado</td><td>Medición cuantitativa mediante algoritmos de comparación de n-gramas entre la salida del modelo y el gold standard.</td><td>Coherencia y Similitud textual</td><td colspan="2">• ROUGE-1
• ROUGE-2
• ROUGE-L
• BLEU score</td></tr>
</table>

¿Qué nivel de desempeño presentan los modelos abstractivos en la generación de resúmenes de resoluciones judiciales de la Corte Superior del Perú?	Evaluar el desempeño de modelos abstractivos en la generación automática de resúmenes de resoluciones judiciales	Existen diferencias en el rendimiento de modelos abstractivos en la generación de resúmenes judiciales.				
¿En qué medida las métricas ROUGE y BLEU permiten evaluar la calidad de los resúmenes generados automáticamente a partir de resoluciones judiciales de la Corte Superior del Perú?	Evaluar la calidad de los resúmenes generados a partir de resoluciones judiciales de la Corte Superior de Lima, Perú utilizando métricas como ROUGE y BLEU	Los modelos desarrollados alcanzan puntuaciones superiores en coherencia, relevancia y concisión (medidas por ROUGE y BLEU) respecto al método base.				
DISEÑO DE INVESTIGACIÓN:	POBLACIÓN Y MUESTRA:		TÉCNICAS E INSTRUMENTOS:		MÉTODO DE ANÁLISIS DE DATOS:	
Tipo: Aplicada **Nivel**: Explicativo - comparativo	**Población**: Conjunto de resoluciones judiciales emitidas por la Corte Superior de Lima, Perú en diversas materias legales en el periodo 2021 al 2022. **Muestra**: Corpus especializado de 6,848 resoluciones judiciales emitidas por las Cortes Superiores de Justicia del Perú.		**Técnicas**: - Extracción y preprocesamiento de texto (limpieza, segmentación). - Análisis comparativo de resúmenes generados vs. Resúmenes de referencia. **Instrumentos**: - Corpus de resoluciones judiciales (base de datos estructurada). - Herramientas de PLN: NLTK, SpaCy, Transformers. **Métricas de evaluación**: ROUGE-1, ROUGE-2. ROUGE-L y BLEU.		**Cuantitativo comparativo**: Se miden los puntajes ROUGE y BLEU obtenidos por cada modelo. Comparación estadística pretest-postest	

FORMATO DE AUTORIZACIÓN DE PUBLICACIÓN EN EL REPOSITORIO INSTITUCIONAL UNAM

Formulario de autorización para la publicación electrónica en la página web del Repositorio Institucional de la Universidad Nacional de Moquegua, de conformidad con la Ley N° 30035 Ley que regula el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto y Resolución de Comisión Organizadora N° 041-2022-UNAM que aprueba el Reglamento que regula el Repositorio Institucional Digital de la UNAM.

1. DATOS DEL AUTOR

Nombres y Apellidos : Diana Carolina Sucapuca Mamani.....
DNI/Carnet de Extranjería/Pasaporte N° : 74277930.....
Correo electrónico : diana.sucapuca.mamani@gmail.com.....
Teléfono/Celular : 979037150.....

2. GRADO, TITULO O ESPECIALIZACIÓN (Marque con una X)

Bachiller	Título Profesional	Magister	Doctor	Segunda Especialidad
	X			

3. TIPO DE PUBLICACIÓN (Marque con una X)

a) Articulo ( )
b) Tesis de pregrado (X)
c) Tesis de Maestría ( )
d) Tesis de Doctorado ( )
e) Monografía ( )
f) Reporte de caso científico o hallazgos ( )
g) Libro ( )
h) Parte de un libro ( )
i) Revisión ( )
j) Conferencia ( )
k) Presentación ( )
l) Artículo preliminar ( )
m) Articulo antes de ser publicado ( )
n) Artículo de prensa ( )
o) Documentos internos ( )
p) Imagen ( )
q) Documentos técnicos ( )
r) Ponencias presentadas en eventos científicos (Jornada, Congreso, Simposio Conferencias, etc.) ( )
s) Informes de suficiencia profesional ( )
t) Patentes producidas por académicos/as ( )

4. TIPO DE TRABAJO DE INVESTIGACIÓN (Marque con una X)

a) Tesis **(X)**
b) Trabajo de Investigación **( )**
c) Trabajo de suficiencia profesional **( )**
d) Trabajo académico **( )**

5. DATOS DE LA PUBLICACION

a) Título : GENERACIÓN AUTOMÁTICA DE RESÚMENES DE RESOLUCIONES JUDICIALES DE LA CORTE SUPERIOR DE JUSTICIA DEL PERÚ MEDIANTE TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL.
b) Campo del conocimiento OCDE (Indicar el código de clasificación OCDE de la obra : https://catalogos.concytec.gob.pe/vocabulario/ocde_ford.html) : **2.02.04**.....
c) Fecha de publicación : ..19..... junio del 2026.....

6. DATOS DEL ASESOR:

a) Nombres y Apellidos : Hugo Euler Tito Chura.....
b) DNI/Pasaporte/Cédula : ...01315850.....
c) ORCID : ...0000-0001-8269-0655.....

FIRMA : .....

DOCUMENTO DE IDENTIDAD : ...74277930.....
NOMBRES Y APELLIDOS : ...Diana Carolina Sucapuca Mamani.....

LICENCIAS

Licencia Estándar

Bajo los siguientes términos, autorizo el depósito de mi trabajo de investigación en el Repositorio Institucional Digital de la UNAM.

Con la autorización de depósito da mi trabajo de investigación, otorgo a la Universidad Nacional de Moquegua, una Licencia No Exclusiva para reproducir. distribuir, comunicar al público, transformar (únicamente mediante su traducción a otros idiomas) y poner a disposición del público mi trabajo de investigación (incluido el resumen), en formato físico o digital, en cualquier medio, conocido o por conocerse, a través de los diversos servicios provistos por la Universidad, creados o por crearse, tales como el Repositorio Institucional Digital de la UNAM, Colección de Tesis, entre otros, en el Perú y en el extranjero, por el tiempo y veces que considere necesarias, y libre de remuneraciones.

En virtud de dicha Licencia, la UNIVERSIDAD NACIONAL DE MOQUEGUA, podrá reproducir mi trabajo de investigación en cualquier tipo de soporte y en más de un ejemplar, sin modificar su contenido, solo con los propósitos de seguridad, respaldo y preservación.

Declaro que el trabajo de investigación es una creación de mi autoría y exclusiva titularidad, o coautoría con titularidad compartida, y me encuentro facultado a conceder la presente licencia y, asimismo, garantizo que dicho trabajo de investigación no infringe derechos de autor de terceras personas.

La UNIVERSIDAD NACIONAL DE MOQUEGUA consignara el nombre del/los autor/es del trabajo de investigación, y no le hará ninguna modificación más de la permitida en la presente licencia.

Autorizo el depósito (marque con una X).

(__X__) **SI autorizo que se deposite inmediatamente.**

FIRMA : .....

DOCUMENTO DE IDENTIDAD : 74277930.....

NOMBRES Y APELLIDOS : Diana Carolina Sucapuca Mamani.....

FECHA : 19 de junio del 2026.....



ANEXO VIII

DECLARACIÓN JURADA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN / TESIS / TRABAJO DE SUFICIENCIA PROFESIONAL

Yo, Diana Carolina Sucapuca Mamani ,
bachiller de la Escuela Profesional de:
Ingeniería de sistemas e informática ,
de la Universidad Nacional de Moquegua, identificado(a) con DNI Nº... 74277930 , autor(a) del Trabajo de Investigación / Tesis / Trabajo de suficiencia profesional denominado:
GENERACIÓN AUTOMÁTICA DE RESÚMENES DE RESOLUCIONES JUDICIALES DE LA CORTE SUPERIOR DE JUSTICIA DEL PERÚ MEDIANTE TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL

Declaro bajo juramento que:

1. El trabajo de investigación es auténtico, siendo resultado de mi trabajo personal, que no se ha copiado, que no se ha utilizado ideas, formulaciones, citas integrales e ilustraciones diversas, sacadas de cualquier tesis, obra, artículo, memoria, etc., (en versión digital o impresa), sin mencionar de forma clara y exacta su origen o autor, tanto en el cuerpo del texto, figuras, cuadros, tablas u otros que tengan derechos de autor. Por tanto, el trabajo de investigación no ha sido plagiado total ni parcialmente.
2. El trabajo de investigación no ha sido autoplagiada, es decir, no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
3. Los datos presentados en los resultados son reales, no han sido falseados, ni duplicados, ni copiados y por tanto los resultados que se presentan en el trabajo de investigación constituirán aporte a la realidad investigada.

Soy consciente de que, el hecho de no respetar los derechos de autor y hacer plagio, es objeto de sanciones universitarias y/o legales, por lo que asumo cualquier responsabilidad que pudiera derivarse de irregularidades en el trabajo de investigación, así como de los derechos sobre la obra presentada.

Moquegua, 19 de junio del 2026



Huella digital

Nombres y Apellidos: Diana Carolina Sucapuca Mamani
DNI: 74277930

Firmado digitalmente por TITO CHURA Hugo Euler FAU 20449347448 soft
Motivo: Soy el autor del documento
Fecha: 24.04.2026 08:42:52 -05:00

Hugo Euler TITO CHURA

DIANA_SUCAPUCA_INFORME_DE_TESIS.pdf

Observaciones Tesis Diana

Tesis 2026

Universidad Nacional de Moquegua

Detalles del documento

Identificador de la entrega
trn:oid:::1:3548967260

Fecha de entrega
24 abr 2026, 8:01 a.m. GMT-5

Fecha de descarga
24 abr 2026, 8:35 a.m. GMT-5

Nombre del archivo
DIANA_SUCAPUCA_INFORME_DE_TESIS.pdf

Tamaño del archivo
2.2 MB

130 páginas

26.457 palabras

169.773 caracteres

3% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 15 palabras)

Fuentes principales

2% Fuentes de Internet
1% Publicaciones
2% Trabajos entregados (trabajos del estudiante)

Fuentes principales

2% Fuentes de Internet
1% Publicaciones
2% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1	Trabajos del estudiante	Asia Pacific Instutute of Information Technology <1%
2	Trabajos del estudiante	Universidad Nacional de Moquegua <1%
3	Trabajos del estudiante	Universidad Internacional de la Rioja <1%
4	Trabajos del estudiante	Universitat Politècnica de València <1%
5	Trabajos del estudiante	Universidad TecMilenio <1%
6	Internet	repositorio.unam.edu.pe <1%
7	Internet	www.doc.ic.ac.uk <1%
8	Internet	repositorio.unah.edu.pe <1%
9	Internet	repositorio.unsch.edu.pe <1%
10	Trabajos del estudiante	ucb <1%
11	Internet	myscale.com <1%

12 Internet
repositorio.ucv.edu.pe <1%

13 Internet
www.raisingawesome.site <1%

14 Trabajos del estudiante
Instituto Tecnológico y de Estudios Superiores de Occidente <1%

15 Trabajos del estudiante
Universitat Oberta de Catalunya <1%

16 Internet
blog.madrigan.com <1%

17 Internet
ebuah.uah.es <1%

18 Internet
repository.library.carleton.ca <1%

19 Internet
rodo.guluarte.com <1%

20 Internet
johal.in <1%

21 Trabajos del estudiante
University Politehnica of Bucharest <1%

22 Internet
pirhua.udep.edu.pe <1%

23 Internet
repositorio.ulc.edu.pe <1%

Universidad Nacional de Moquegua

RESOLUCIÓN DE CONSEJO UNIVERSITARIO
N° 570-2026-CU

Moquegua, 26 de agosto de 2026

VISTOS:

La Resolución de Consejo de Facultad N° 410-2026-CF-FAIN-UNAM, emitida por la Decana de la Facultad de Ingenierías; el Informe N° D0451-2026-UNAM-R-FAIN, emitido por la Decana de la Facultad de Ingenierías, el Informe Técnico N° D0123-2026-SG-GT, emitido por la Especialista Administrativo II de Grados y Títulos; el Informe N° D0437-2026-UNAM-R-SG, emitido por el Secretario General de la UNAM; el Proveído N° 4234-2026-R, emitido por el señor Rector; el Acuerdo del Pleno del Consejo Universitario de Sesión Extraordinaria N° 35-2026-SE-CU-UNAM, de fecha 25 de agosto de 2026; y,

CONSIDERANDO:

Que, conforme al cuarto párrafo del artículo 18 de la Constitución Política del Perú, dispone que cada Universidad es autónoma en su régimen normativo, de gobierno, académico, administrativo y económico. Y el artículo 8° de la Ley N° 30220-Ley Universitaria, establece que la referida autonomía inherente a las Universidades, se ejerce de conformidad con la Constitución y demás normativa aplicable. Concordante con el artículo 1, de la Ley N° 31520, Ley que restablece la autonomía y la institucionalidad de las universidades peruanas, así como la Ley N° 32141, Ley que modifica la Ley N° 30220-Ley Universitaria, a fin de fortalecer la institucionalidad de las Universidades Públicas licenciadas;

Que, el artículo 58 de la Ley N° 30220-Ley Universitaria, establece que el Consejo Universitario es el máximo órgano de gestión, dirección y ejecución académica y administrativa de la Universidad. Concordante con el artículo 18° del Estatuto de la Universidad Nacional de Moquegua, aprobado bajo Resolución de Asamblea Universitaria N° 015-2025-AU-UNAM;

Que, el artículo 59 de la Ley Universitaria-Ley N° 30220, establece las atribuciones del Consejo Universitario, y específicamente el numeral 59.9 establece; Conferir los grados académicos y los títulos profesionales aprobados por las Facultades y Escuela de Posgrado, así como otorgar distinciones honoríficas y reconocer y revalidar los estudios, grados y títulos de universidades extranjeras, cuando la universidad está autorizada por la Superintendencia Nacional de Educación Superior Universitaria. Concordante con el numeral 14, del artículo 20° del Estatuto de la Universidad Nacional de Moquegua;

Que, mediante Resolución de Asamblea Universitaria N° 006-2026-AU, de fecha 20 de abril de 2026, se aprobó la incorporación de la Disposición Transitoria N° 22 al Estatuto de la Universidad Nacional de Moquegua, referida a la excepción para la emisión de los grados académicos de bachiller y títulos profesionales consignando la denominación de "Facultad de Ingeniería y Arquitectura";

Que, mediante la Resolución de Consejo de Facultad N° 410-2026-CF-FAIN-UNAM, de fecha 13 de agosto de 2026, el Consejo de Facultad de Ingenierías aprobó el otorgamiento del Título Profesional de Ingeniero de Sistemas e Informática, por la modalidad de Tesis, a un (01) Bachiller de la Escuela Profesional de Ingeniería de Sistemas e Informática de la Facultad de Ingeniería y Arquitectura de la Universidad Nacional de Moquegua, de acuerdo al siguiente detalle: DIANA CAROLINA SUCAPUCA MAMANI;

Que, mediante Informe N° D0451-2026-UNAM-R-FAIN, de fecha 17 de agosto de 2026, la Decana de la Facultad de Ingenierías, remite a Secretaría General la Resolución de Consejo de Facultad N° 410-2026-CF-FAIN-UNAM, de fecha 13 de agosto de 2026;

Que, mediante Informe Técnico N° D0123-2026-SG-GT de fecha 19 de agosto de 2026, e Informe N° D0437-2026-UNAM-R-SG de fecha 20 de agosto de 2026, el Secretario General de la UNAM, remite al Rectorado el expediente para el trámite de Título Profesional de Ingeniero de Sistemas e Informática a favor de la Bachiller: DIANA CAROLINA SUCAPUCA MAMANI de la Escuela Profesional de Ingeniería de Sistemas e Informática, quien se encuentra apta para la expedición del Título Profesional de Ingeniero de Sistemas e Informática, al haber cumplido satisfactoriamente con los requisitos legales y reglamentarios, exigidos en el Reglamento de Grados y Títulos de la Universidad Nacional de Moquegua, razón por la cual, el señor Secretario General incluye dicho expediente como punto de agenda para ser tratado en Sesión Extraordinaria de Consejo Universitario de fecha 25 de agosto de 2026;

Que, el artículo 44° de la Ley N° 30220, Ley Universitaria señala que las universidades otorgan los grados académicos de Bachiller, Maestro, Doctor y los Títulos Profesionales que correspondan, a nombre de la Nación (...). Asimismo, la Ley N° 31803, modifica el artículo en mención adicionando; La Superintendencia Nacional de Educación Superior Universitaria (SUNEDU) determina los criterios, así como establece y supervisa el procedimiento administrativo para el reconocimiento de grados y títulos otorgados en el extranjero, en el marco de la normativa vigente;

Que, el numeral 45.2 del artículo 45° de la Ley N° 30220, Ley Universitaria, precisa los requisitos mínimos para la obtención del título profesional, como son: el grado de Bachiller y la aprobación de una tesis o trabajo de suficiencia profesional. Asimismo, establece que el título profesional solo se puede obtener en la universidad en la cual se haya obtenido el grado de bachiller;

Domicilio Legal: Prolongación Av., Ancash S/N - Moquegua
RUC: 20449347448 - Teléfono: (053) 670168
Página Web: www.unam.edu.pe

Universidad Nacional de Moquegua

RESOLUCIÓN DE CONSEJO UNIVERSITARIO
N° 570-2026-CU

Moquegua, 26 de agosto de 2026

Que, mediante Resolución de Comisión Organizadora N° 1153-2021-UNAM, se aprobó el Reglamento de Grados y Títulos de la Universidad Nacional de Moquegua, el mismo que fue modificado a través de las Resoluciones de Comisión Organizadora N° 410; 453 y 890-2022, N° 880-2023-UNAM y N° 081, 484-2024-UNAM el cual establece los requisitos y norma los procedimientos para la obtención de los grados académicos y títulos profesionales conferidos por esta Casa Superior de Estudios;

Que, bajo el amparo de los considerandos precedentes, el Pleno del Consejo Universitario en Sesión Extraordinaria N° 35-2026-SE-CU-UNAM, de fecha 25 de agosto de 2026, emitió el siguiente acuerdo: POR UNANIMIDAD: CONFERIR, a nombre de la Nación el Título Profesional de Ingeniero de Sistemas e Informática por la modalidad de Tesis a la Bachiller: DIANA CAROLINA SUCAPUCA MAMANI, de la Escuela Profesional de Ingeniería de Sistemas e Informática, de la Facultad de Ingeniería y Arquitectura de la Universidad Nacional de Moquegua;

En uso de las facultades y atribuciones conferidas por el artículo 18 de la Constitución Política del Perú, la Ley N° 30220 Ley Universitaria, el Estatuto de la Universidad Nacional de Moquegua, y a lo acordado por el pleno del Consejo Universitario;

SE RESUELVE:

ARTÍCULO 1°: CONFERIR, A NOMBRE DE LA NACIÓN, a la Bachiller

- **DIANA CAROLINA SUCAPUCA MAMANI**

El **TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS E INFORMÁTICA.**

Por la modalidad de TESIS, de la Escuela Profesional de Ingeniería de Sistemas e Informática, de la Facultad de Ingeniería y Arquitectura de la Universidad Nacional de Moquegua.

ARTÍCULO 2°: ENCARGAR, al Secretario General de la UNAM, para que realice las acciones pertinentes relacionadas con la inscripción correspondiente en el Registro de Grados y Títulos de la Superintendencia Nacional de Educación Superior Universitaria - SUNEDU.

ARTÍCULO 3°: DISPONER, el cumplimiento de la presente Resolución a todas las dependencias de la Universidad Nacional de Moquegua.

ARTÍCULO 4°: DISPONER la publicación de la presente resolución en el Portal Institucional de la Universidad Nacional de Moquegua www.unam.edu.pe

REGÍSTRESE, PUBLÍQUESE Y CÚMPLASE;

UNIVERSIDAD NACIONAL DE MOQUEGUA SECRETARIA GENERAL

UNIVERSIDAD NACIONAL DE MOQUEGUA CONSEJO UNIVERSITARIO

MTRO. JOSE LUIS PACHECO CACERES
SECRETARIO GENERAL

DR. HUGO RUBEN MARCA MAQUERA
RECTOR
PRESIDENTE DEL CONSEJO UNIVERSITARIO

DISTRIBUCIÓN: VRA, VRI, FAIN, EPISI, GyT, DASA, OTI, *Interesada*, C.c. Arch.
SVR

Domicilio Legal: Prolongación Av., Ancash S/N - Moquegua
RUC: 20449347448 - Teléfono: (053) 670168
Página Web: www.unam.edu.pe