UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Reconocimiento y extracción de entidades a partir de archivos del pasado reciente

Informe de Proyecto de Grado presentado por

Natalie Valentina Alaniz Ferreira, Facundo Díaz Rodríguez, Agustín Matías Martínez Acuña

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Aiala Rosá
Ignacio Sastre

Montevideo, 2025-11-28

Agradecimientos

Agradecemos, en primer lugar, a Aiala Rosá e Ignacio Sastre por guiarnos y acompañarnos a lo largo del proyecto. Agradecemos su buena disposición para compartir sus conocimientos y experiencia a lo largo de nuestra trayectoria académica y por ser nuestros referentes.

Durante nuestras carreras, muchos docentes y divulgadores dedicaron su tiempo, esfuerzo y vocación a despertar nuestra curiosidad e interés por el conocimiento. A todos ellos, un sincero agradecimiento.

A nuestros amigos y compañeros de estudio, gracias por compartir con nosotros muchos momentos y experiencias que nos acompañarán en las etapas que vendrán. En su compañía, a lo largo de los cursos, en las horas de estudio compartidas, en las charlas, en las vacaciones y en las salidas, encontramos la motivación para seguir adelante. Transitar la carrera con ustedes fue un verdadero placer y un privilegio.

Un especial agradecimiento a nuestras familias que, desde el primer día hasta el último, han estado presentes apoyándonos tanto en los momentos altos como en los bajos. A ellos les dedicamos este trabajo, que marca el fin de esta hermosa (y larga) etapa de nuestras vidas. Este logro también es suyo.

Reservamos también un agradecimiento al equipo de *Cluster.uy*, cuya infraestructura nacional y pública hizo posible el desarrollo de este proyecto y la ejecución de los experimentos. También a sus administradores, que nos perdonaron por estar pasados de la cuota en 3 TB.

¡Muchas gracias a todas y todos!

Resumen

Este proyecto de grado desarrolla un sistema para el reconocimiento y extracción de información (IE) a partir de documentos históricos del período dictatorial y predictatorial uruguayo, con el objetivo de aportar a los esfuerzos de memoria, verdad y justicia. La iniciativa se enmarca en el proyecto CRUZAR[1], que busca sistematizar, organizar y permitir el análisis de más de dos millones de documentos digitalizados provenientes del Archivo Berrutti y otros repositorios oficiales.

El trabajo se centra en las dos primeras etapas del proceso de IE:

Reconocimiento Óptico de Caracteres (OCR): Consiste en obtener transcripciones a partir de imágenes. Existen dificultades particulares del contexto: los documentos presentan zonas dañadas, disposiciones complejas y artefactos que dificultan la transcripción. Para esto se evaluaron LLMs multimodales, explorando su capacidad para mejorar la calidad de transcripción de los documentos. Además, se implementa una etapa de preprocesamiento que logra disminuir el tiempo de transcripción por documento y aumentar su calidad.

Reconocimiento de Entidades Nombradas (NER): Esta tarea consiste en identificar y clasificar entidades mencionadas en los textos, como personas, lugares u organizaciones. Para esto se entrenaron modelos basados en BERT mediante ajuste fino (*finetuning*), adaptándolos a las particularidades del dominio. En muchos casos, este tipo de entrenamiento requiere una gran cantidad de datos etiquetados. Dado el tamaño reducido del conjunto de datos, se recurrió a LLMs para generar ejemplos sintéticos que imitan los existentes. Para aproximar mejor el contexto real, se añadió un módulo de ruido que simula los errores habituales de OCR. Finalmente, se comparó el desempeño de estrategias de entrenamiento tanto integradas como secuenciales, evaluándose mediante métricas de precisión, *recall* y $F1_{\text{macro}}$. Como resultado de la experimentación, para OCR se obtuvieron transcripciones de alta calidad respecto a las disponibles anteriormente, incluso en escenarios de imágenes dañadas o ruidosas, donde la herramienta por defecto muestra un rendimiento muy bajo. Esta última alcanzó valores de BLEU de 0,68, 0,22 y 0,05 en documentos clasificados como de fácil, media o difícil transcripción, respectivamente. Mientras que utilizando LLMs multimodales se obtuvieron

[1] https://cruzar.edu.uy/

resultados superiores: BLEU de 0,85, 0,72 y 0,45 en los mismos conjuntos.

En el caso de NER, el modelo BERT logra alcanzar 0,51 en $F1_{\text{macro}}$, un aumento significativo respecto al máximo alcanzado por proyectos anteriores ($F1_{\text{macro}}$ de 0,26). El uso de datos sintéticos logra una mejora de $F1_{\text{macro}}$ de 3 % en el conjunto de test.

En conclusión, las mejores herramientas construidas superan al estado del arte actual de CRUZAR en ambas tareas, mostrando las capacidades de los modelos generativos generales para tareas específicas de IE en este dominio.

A futuro, es posible seguir trabajando en modelos que mejoren la calidad de la transcripción o de la extracción de entidades, experimentando nuevos enfoques o mejorando la calidad de los ejemplos sintéticos. Por otro lado, los resultados actuales permiten continuar hacia etapas siguientes, como puede ser la extracción de eventos, la resolución de correferencias o la generación de un grafo de conocimiento que ayude a explorar patrones en los textos.

Palabras clave: PLN, NER, OCR, Generación de ejemplos sintéticos, Proyectos de Grado, Computación

Índice general

Capítulo 1

Introducción

Este trabajo es una contribución al proyecto CRUZAR[1], al esfuerzo colectivo para transcribir textos del período dictatorial y predictatorial uruguayo, así como a la búsqueda de respuestas sobre las causas, consecuencias y responsabilidades de las violaciones a los derechos humanos cometidas en esos períodos.

1.1. Proyectos de recuperación de la memoria histórica

En 2006, el Estado uruguayo recuperó un archivo de inteligencia militar que se encontraba oculto en el ex *Centro General de Instrucción para Oficiales de Reserva* (CGIOR), actual *Centro de Altos Estudios Nacionales* (CALEN). Esta acción fue impulsada por la *Ministra de Defensa Nacional Dra. Azucena Berrutti*, de quien el archivo recibe su denominación (Sitios de Memoria Uruguay, 2025).

Este fue el más reciente de una serie de hallazgos que completan una colección masiva de documentos militares que abarcan el período de 1968 a 2004. La posterior digitalización de la colección permitió realizar tareas de transcripción a formato de texto mediante herramientas de reconocimiento óptico de caracteres (OCR).

Los esfuerzos computacionales para obtener información de estos documentos se sintetizan en *CRUZAR*, un proyecto de sistematización de información de archivos del pasado reciente vinculados al terrorismo de Estado y a violaciones a los derechos humanos. Su objetivo es el ordenamiento y la clasificación del material, así como la elaboración de un programa informático que permita cruzar la información contenida en esos archivos.

[1] https://cruzar.edu.uy/

El proyecto cuenta con un repositorio de datos con más de dos millones de documentos; el origen de la mayoría de ellos es el ya mencionado *Archivo Berrutti*, compuesto por unos 1500 rollos de *microfilm* (ver Figura 1.1), cada uno de miles de páginas en promedio.

Figura 1.1: Rollo de *microfilm*[2], formato de almacenamiento analógico que conserva documentos miniaturizados en una película fotográfica de alta durabilidad.

El repositorio contiene un universo heterogéneo de archivos y documentos militares, así como los *microfilms* producidos por la *Dirección Nacional de Información e Inteligencia* (DNII), el *Órgano Coordinador de Operaciones Antisubversivas* (OCOA) y el *Servicio de Información de Defensa* (SID). En dichos rollos se encuentran: actas de personas detenidas, reportes de allanamientos, escuchas telefónicas, comunicados oficiales, expedientes administrativos, expedientes y fichas de personas. Asimismo, contiene memorandos, recortes de prensa, solicitudes de información, órdenes de captura, transcripciones y reproducciones de publicaciones incautadas, entre otros (Facultad de Información y Comunicación y Facultad de Ingeniería, Universidad de la República, 2025).

Calidad de los documentos

Los *microfilms* recuperados se encontraban almacenados en condiciones inadecuadas, potencialmente expuestos durante años a factores como la humedad, temperaturas variables y escasa o nula ventilación. Estas condiciones propiciaron el deterioro de las imágenes, lo que implicó la pérdida parcial o incluso total de la legibilidad en algunos casos (Facultad de Información y Comunicación y Facultad de Ingeniería, Universidad de la República, 2025).

Las herramientas de OCR se ven fuertemente afectadas por la calidad de la imagen, lo que puede degradar significativamente la transcripción final.

[2]Imagen de Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Positive_roll_film.jpg

La necesidad de acceder a datos de referencia para el entrenamiento de modelos motivó el desarrollo del sistema *Leyendo Unidos para Interpretar loS Archivos* (LUISA[3]), donde personas colaboran transcribiendo manualmente recortes de documentos de baja calidad. El usuario visualiza un recorte específico del documento y escribe lo que interpreta; luego, el sistema compara múltiples transcripciones del mismo fragmento para validar el texto final y mejorar el reconocimiento automatizado. Varios usuarios hacen lo mismo y el sistema une las respuestas para recuperar el texto original; estos fragmentos son lo suficientemente pequeños como para no revelar información sensible.

El uso de la herramienta logró que 1500 páginas del archivo fueran procesadas, mejorando sensiblemente la calidad con respecto a las transcripciones iniciales, generando un conjunto de datos. Paralelamente, se utilizan herramientas automáticas para generar conjuntos de datos.

Una de las líneas de investigación y desarrollo de CRUZAR y de nuestro proyecto es la recopilación, puesta a punto y construcción de herramientas para obtener transcripciones de mejor calidad de forma automática.

[3] https://mh.udelar.edu.uy/luisa/

1.2. Nuestro proyecto

Este proyecto de grado busca implementar un sistema dedicado a la extracción de información a partir de imágenes. En este contexto, se trabaja en dos tareas: el reconocimiento óptico de caracteres (OCR) y el reconocimiento de entidades nombradas (NER).

Para la tarea OCR se evalúa el rendimiento de grandes modelos de lenguaje (LLM) multimodales. Si bien los métodos tradicionales de OCR incorporan componentes lingüísticos (por ejemplo, a través de diccionarios), el uso de LLMs busca utilizar información contextual del texto en el proceso de transcripción. Aunque esto representa una mejora significativa en la legibilidad del texto, pueden surgir problemas relacionados con las alucinaciones del modelo, donde se agrega información que no está presente. Este riesgo es preocupante al trabajar con datos sensibles, ya que cualquier error podría comprometer la integridad de la información.

Para la tarea NER se utilizan modelos de lenguaje, realizando *finetuning* de modelos preentrenados. Además, se generan datos sintéticos con el fin de complementar el conjunto de datos reales, que se conforma de un número limitado de ejemplos etiquetados, lo cual dificulta el entrenamiento.

Todos los experimentos se ejecutan en infraestructura proporcionada por el *Centro Nacional de Supercomputación* (*ClusterUY*[4]) con las siguientes características:

- **CPU**: `AMD EPYC 7763`[5].
  - #**Núcleos**: `128`.
- **GPU**: `NVIDIA A40`[6].
  - **VRAM**: `48 GB GDDR6`.
- **RAM**: `256 GB`.
- **Almacenamiento secundario**: `300 GB SSD`.
- **OS**: `GNU/Linux CentOS 7`.

[4] https://cluster.uy/
[5] https://www.amd.com/es/products/processors/server/epyc/7003-series/amd-epyc-7763.html
[6] https://www.nvidia.com/es-la/data-center/a40/

1.2.1. Objetivos

Con el fin de abordar de manera sistemática los aspectos centrales del tema, se plantean objetivos generales y específicos que orientarán el desarrollo del proyecto:

General

- Desarrollo de herramientas para el reconocimiento de texto y extracción de entidades a partir de documentos que integran el archivo del proyecto *CRUZAR*.

Específicos

I. Estudio de bibliografía sobre extracción de información, enfoques clásicos y recientes.

II. Evaluar técnicas de OCR basadas en grandes modelos del lenguaje multimodales para su aplicación en documentos históricos deteriorados.

III. Recopilar y generar un corpus para el entrenamiento y la evaluación de NER.

IV. Utilizar LLMs para generar datos sintéticos que aumenten el conjunto de entrenamiento de NER y evaluar su calidad y desempeño.

V. Ajustar modelos de lenguaje de tipo BERT para la tarea de NER. Explorar diferentes técnicas y métodos de entrenamiento y su impacto en el rendimiento de los modelos.

VI. Diseñar e implementar un prototipo modular de *pipeline* que integre las etapas de OCR y NER.

1.2.2. Organización del documento

En este informe se reporta la totalidad del desarrollo del proyecto; esto incluye análisis, diseño, experimentación y desarrollo. Consta de 7 capítulos con su siguiente desglose:

Capítulo I: Se presenta una introducción general, contextualizando el propósito y alcance del trabajo.

Capítulo II: Corresponde al primer artefacto del proyecto, donde se presenta un marco teórico y un estudio del estado del arte relevante al trabajo realizado.

Capítulo III: Se estudian los antecedentes del proyecto CRUZAR.

Capítulo IV: Se profundiza en el problema a resolver y se presenta una arquitectura que permite integrar los módulos de extracción de información desarrollados en el presente proyecto.

Capítulo V: Se exploran técnicas de OCR basadas en LLMs multimodales. Las mismas son evaluadas y comparadas con métodos tradicionales.

Capítulo VI: Se presenta el proceso de entrenamiento y evaluación de modelos de reconocimiento de entidades nombradas, basados en la arquitectura BERT. Se explora el uso de LLMs para la generación de datos sintéticos con el fin de suplir la escasez.

Capítulo VII: Se detallan las conclusiones del proyecto desarrollado y las posibles mejoras a futuro.

Capítulo 2

Marco Teórico / Estado del arte

Este capítulo está dedicado a introducir una noción general de los conceptos teóricos utilizados en el proyecto, así como una revisión del estado del arte, incluyendo las principales arquitecturas, modelos y metodologías que son actualmente utilizados en sistemas de extracción de información.

La tarea de NER es intrínsecamente lingüística, mientras que el OCR parte como un problema de reconocimiento de imágenes que se beneficia de la introducción de componentes lingüísticos.

Por otro lado, los grandes modelos de lenguaje buscan modelar la distribución de probabilidad del lenguaje natural. En años recientes, han visto un acelerado desarrollo y aplicación a múltiples tareas, incluyendo las anteriormente mencionadas.

En primer lugar, se introducen los modelos de lenguaje, que constituyen la base de los distintos enfoques empleados a lo largo del proyecto (OCR, generación de datos sintéticos y NER). A continuación, se describe la arquitectura *Transformer* a partir de la cual se desarrollan los modelos de lenguaje utilizados en este proyecto.

Se introducen los LLMs multimodales, utilizados en este proyecto para la resolución de la tarea OCR. Posteriormente, se exploran técnicas de ajuste y optimización de LLMs, con herramientas que facilitan la búsqueda de hiperparámetros.

Se profundiza en la tarea de OCR, describiendo sus principales etapas y los fundamentos tradicionalmente utilizados en la implementación de modelos como *Tesseract*. Se mencionan alternativas basadas en el uso de LLMs. Además, se definen las principales métricas para evaluar el rendimiento en esta tarea.

Finalmente, se describen las principales técnicas utilizadas para resolver la tarea NER, incluidos los métodos basados en modelos de lenguaje. También se presentan las principales métricas utilizadas para la evaluación de modelos en esta tarea. En este contexto, se exploran técnicas de generación de datos sintéticos, tanto mediante alternativas deterministas basadas en diccionarios como mediante soluciones estocásticas basadas en LLMs.

2.1. Modelos de Lenguaje

Un modelo de lenguaje es un modelo de aprendizaje automático capaz de predecir la siguiente palabra de una tira de texto. Formalmente, un modelo de lenguaje asigna una probabilidad a cada posible palabra siguiente o, equivalente, proporciona una distribución de probabilidad sobre las posibles continuaciones. También pueden asignar una probabilidad a una oración completa (Jurafsky y Martin, 2025).

Los ejemplos más sencillos de estos modelos son los basados en n-gramas, que se definen como modelos probabilísticos que utilizan $n-1$ palabras para predecir la n-ésima ocurrencia.

En definitiva, se encargan de calcular $P(w|h)$: la probabilidad de que la siguiente palabra sea w, dada la secuencia (o historial) h. Una forma de estimar la probabilidad puede consistir en calcular la frecuencia relativa a partir de un conjunto de datos. Por ejemplo:

$$P(\text{soleado} \mid \text{El día está}) = \frac{\#(\text{El día está soleado})}{\#(\text{El día está})}$$

Siendo $\#(X)$ la cantidad de ocurrencias de la secuencia X en los datos de entrenamiento.

Tokenización

Un modelo de lenguaje debe considerar y asignar una probabilidad a todas las posibles formas de continuar la secuencia sobre la que trabaja. Un modelo basado en n-gramas realiza predicciones a nivel de palabra; sin embargo, esto requiere considerar todos los elementos del vocabulario utilizado, lo que puede ser de gran tamaño, lo que aumenta los costos de memoria y de cómputo requeridos.

Una forma ampliamente extendida en los enfoques actuales para enfrentar esta problemática es trabajar con representaciones de mayor nivel de granularidad, por ejemplo, a nivel de subpalabra. En este caso, los *tokens* que se aproximan al concepto de morfema.

A modo de ejemplo, es posible reducir el siguiente vocabulario

$$\{\text{salta\_}, \text{saltaba\_}, \text{nada\_}, \text{nadaba\_}, \text{canta\_}, \text{cantaba\_}\}$$

a un conjunto de menor cardinalidad, pero capaz de expresar la misma información.

{salt, nad, cant, a_, aba_}

Un *token* es la unidad básica en la que se segmenta el texto para su procesamiento. Cada uno de ellos no necesariamente corresponde a una palabra o subpalabra, sino que también puede corresponder a símbolos especiales que representan, por ejemplo, espacios o finales de oración.

2.2. Arquitectura *Transformer*

En el PLN, los modelos de lenguaje basados en la arquitectura *Transformer* (Vaswani y cols., 2023) han impulsado avances gracias a su capacidad para manejar eficientemente el contexto y las relaciones semánticas en los textos. Arquitecturas anteriores, como redes neuronales recurrentes (RNN) o redes de memoria a corto y largo plazo (LSTM), procesan los *tokens* de forma secuencial. En cambio, la arquitectura *Transformer* permite procesar simultáneamente todas las posiciones de una secuencia, lo que mejora la eficiencia y la escalabilidad.

La arquitectura *Transformer* consta de dos componentes: el codificador (*encoder*) y el decodificador (*decoder*), como se ilustra en la Figura 2.1. El *encoder* toma como entrada representaciones vectoriales de los *tokens* que componen la secuencia de entrada, denominadas *word embeddings*. Son procesados mediante múltiples capas de autoatención (*self-attention*) y de redes *feed-forward*, generando una nueva secuencia de vectores; cada elemento corresponde a una representación contextualizada del *token* de entrada. Estas representaciones no solo capturan el significado individual de cada *token*, sino que también integran información contextual, lo que permite al modelo desambiguar palabras polisémicas y comprender relaciones sintácticas.

Por otro lado, el *decoder* recibe los *embeddings* originales de los *tokens* $(x_1, \ldots, x_n)$, junto con las representaciones contextualizadas del *encoder*, para generar una nueva secuencia. A modo de ejemplo, en tareas como NER, cada vector generado puede asociarse directamente a etiquetas específicas de los *tokens* originales. En aplicaciones generativas, como en LLMs, el *decoder* predice secuencialmente cada *token*, condicionado por los *tokens* previamente generados y por un *prompt* inicial.

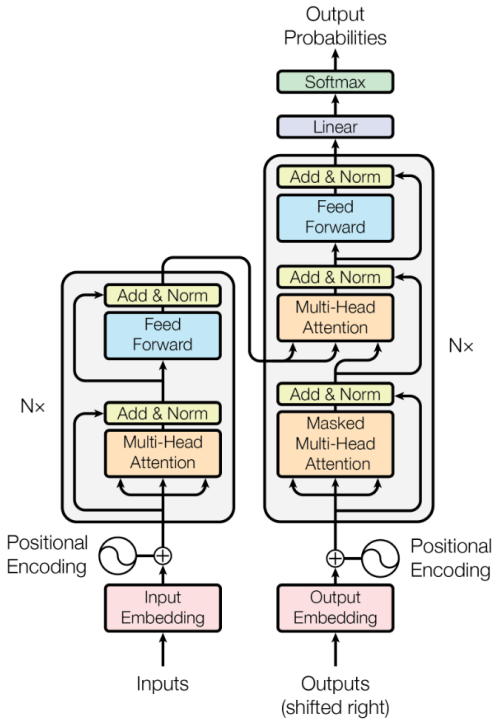


Figura 2.1: Arquitectura *Transformer* (Vaswani y cols., 2023)

Mecanismo de Atención: Self-Attention y Multi-Head Attention

Uno de los componentes clave del *Transformer* es el mecanismo de autoatención (*self-attention*), que permite que cada *token* de la secuencia se relacione con los demás *tokens*. Para lograrlo, cada *token* se representa con tres vectores generados a partir de transformaciones lineales: consultas (*queries*), claves (*keys*) y valores (*values*). El cálculo de atención se realiza como:

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.1}$$

Donde d_k es la dimensión de los vectores clave.

Para mejorar la capacidad de representación del modelo, se utiliza el mecanismo de *Multi-Head Attention*, en el cual múltiples cabezas de atención calculan diferentes representaciones contextuales en paralelo, que luego se combinan en una única salida. Esto permite que el modelo capture múltiples aspectos de la relación semántica entre los *tokens*.

Positional Encoding

Dado que el modelo *Transformer* no tiene una estructura recurrente como las RNNs, se introduce una codificación posicional (*positional encoding*) para capturar información sobre el orden de la secuencia. Esto se logra mediante la adición de un vector a los *word embeddings*, que generalmente utiliza funciones sinusoidales y cosinusoidales. Este método permite que el modelo aprenda la estructura de la secuencia sin recurrir a recurrencias.

2.3. Modelos del Lenguaje basados en *Transformer*

Los modelos de lenguaje basados en transformer, como BERT (Devlin, Chang, Lee, y Toutanova, 2019) o GPT (Radford, Narasimhan, Salimans, y Sutskever, 2018), son modelos de lenguaje con una gran cantidad de parámetros y entrenados sobre corpus textuales de gran tamaño. Como consecuencia de estas características, presentan un gran rendimiento en tareas de comprensión del texto, como el análisis de sentimiento, el reconocimiento de entidades nombradas, la traducción automática y la generación de texto.

Dependiendo del diseño de la arquitectura, estos modelos pueden clasificarse en tres tipos:

- **Decoder-only** (ej. GPT 2, LLAMA): orientado a la generación autorregresiva de texto, prediciendo el siguiente *token* dada una secuencia previa.
- **Encoder-only** (ej. BERT): especializados en tareas de comprensión de lenguaje. Permite generar representaciones contextualizadas de los *tokens* de entrada.
- **Encoder-Decoder** (ej. BART): utilizados para tareas de traducción automática o generación de resúmenes.

2.3.1. *Decoder-Only*: Generative Pre-trained Transformer (GPT)

Generative Pretrained Transformer es una arquitectura utilizada para la generación de texto. El modelo toma como entrada una secuencia de *embeddings* y los procesa de forma causal, generando una representación contextualizada de cada elemento que incorpora información de los elementos anteriores. Posteriormente, los utiliza para generar una distribución de probabilidad sobre el conjunto de *tokens* que pueden ser el próximo elemento de la secuencia. Como consecuencia del uso exclusivo de un mecanismo de atención causal, es posible utilizar este modelo de forma *autorregresiva*, donde el resultado de cada inferencia es agregado a la secuencia para formar una nueva entrada para el modelo, permitiendo así generar una continuación del texto de largo arbitrario.

La elección del siguiente *token* de la secuencia puede realizarse mediante distintas estrategias de decodificación. Cada método representa un compromiso entre diversidad, coherencia semántica, eficiencia y control sobre la salida generada.

Métodos de decodificación

La calidad de la secuencia generada no depende únicamente de la arquitectura del modelo, sino también del método de decodificación. Este proceso consiste en transformar las distribuciones de probabilidad generadas por el modelo en secuencias de texto. A continuación, se describen los principales métodos de decodificación:

- **Decodificación *Greedy*:** La decodificación *greedy* selecciona en cada paso el *token* más probable según la distribución generada por el modelo. Es un método computacionalmente eficiente y determinista, pero puede generar salidas subóptimas a nivel global, al no considerar alternativas con menor probabilidad local que podrían derivar en secuencias globalmente más probables.
- **Decodificación *Beam Search*:**

  El algoritmo de *beam search* generaliza la decodificación *greedy*, manteniendo un conjunto de k secuencias parciales que maximiza la suma de los logaritmos de las probabilidades acumuladas. En cada paso se generan nuevas extensiones posibles para las secuencias activas y se seleccionan las k más prometedoras. Esta técnica mejora la cobertura del espacio de hipótesis, pero a cambio implica un mayor costo computacional. La Figura 2.2 ilustra este proceso.

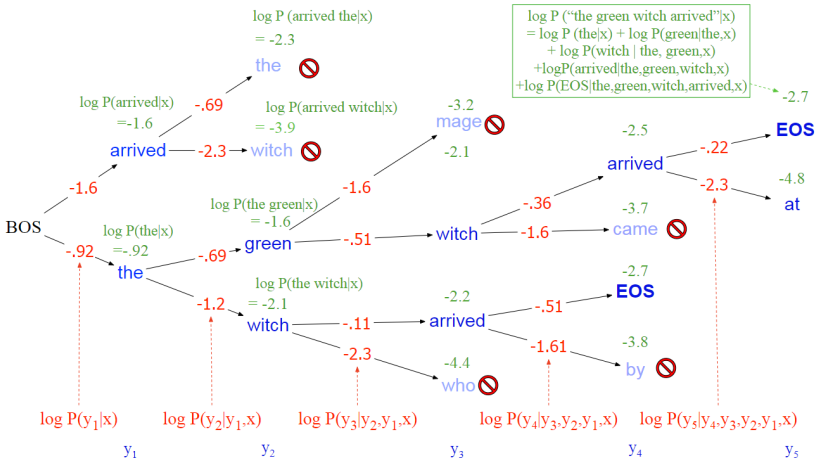


Figura 2.2: Beam Search (Jurafsky y Martin, 2025).

Sampling Estocástico

El muestreo estocástico introduce aleatoriedad en el proceso de generación al seleccionar el siguiente *token* en función de la distribución de probabilidad del modelo. Existen variantes que cambian el equilibrio entre exploración y precisión semántica:

- **Sampling puro**: Se realiza un muestreo de la distribución de *softmax* sin restricciones. Aunque esta estrategia favorece la diversidad, puede generar incoherencias semánticas. Esto se debe a que *softmax* asigna una probabilidad positiva, aunque muy baja, a una gran cantidad de *tokens* irrelevantes. La suma de estas probabilidades individuales, al abarcar todos los *tokens* del vocabulario, se constituye en una porción no despreciable de la masa total de probabilidad. Por ende, al realizar un muestreo sin restricciones, existe una posibilidad no menor de seleccionar un *token* incoherente simplemente por pertenecer a una porción extensa del vocabulario conocida como *long tail*.
- ***Top-k Sampling***: Restringe el espacio de muestreo a los k *tokens* con mayor probabilidad, equilibrando la exploración y la coherencia del espacio de *tokens* a seleccionar.
- ***Top-p Sampling***: Seleccionan el conjunto mínimo de *tokens* cuya probabilidad acumulada supera un umbral p.

Escalado según temperatura

El hiperparámetro de temperatura (τ) modula la entropía de la distribución de salida escalando los *logits* del modelo. Se implementa dividiendo los *logits* por τ antes de aplicar *softmax*.

- $\tau < 1$: concentra la masa de probabilidad en los *tokens* más probables, lo que favorece salidas deterministas.
- $\tau > 1$: suaviza la distribución, incrementando la diversidad, favoreciendo respuestas más creativas.
- $\tau = 1$: conserva la distribución original del modelo.

Técnicas de *prompting*

Un *prompt* es una cadena de texto que se le proporciona al modelo de lenguaje para que resuelva una tarea específica. El *prompting* es el conjunto de técnicas empleadas para diseñar dichas entradas, con el fin de orientar la salida del modelo sin necesidad de modificar sus parámetros. Esta forma de implementar soluciones es relativamente sencilla y poco costosa, ya que permite adaptar un modelo generalista a una tarea concreta sin recurrir a *fine-tuning*.

- ***Zero-shot prompting***: Consiste en pedirle directamente al modelo que resuelva una tarea sin proporcionarle ejemplos de cómo debería hacerlo.

Se espera que, gracias al entrenamiento en grandes conjuntos de datos, el modelo haya adquirido suficiente conocimiento del dominio para responder adecuadamente. Este método es útil para tareas generales, como la clasificación de sentimiento, la respuesta a preguntas o la extracción de información, en las que la formulación de la instrucción puede bastar para guiar al modelo.

- ***few-shot prompting***: Consiste en pedirle directamente al modelo que resuelva una tarea sin proporcionarle ejemplos de cómo debería hacerlo. Se espera que, gracias al entrenamiento en grandes conjuntos de datos, el modelo haya adquirido suficiente conocimiento del dominio para responder adecuadamente. Este método es útil para tareas generales, como la clasificación de sentimiento, la respuesta a preguntas o la extracción de información, en las que la formulación de la instrucción puede bastar para guiar al modelo.
- ***Chain of Thought prompting***: Es una extensión de la técnica anterior; se busca que el modelo produzca pasos de razonamiento explícitos antes de generar la respuesta final. En lugar de limitarse a dar una salida directa, el modelo genera una explicación paso a paso que le permite estructurar mejor la solución y reducir errores en tareas complejas. Al incluir ejemplos en el *prompt* que detallen la secuencia de razonamientos intermedios, los LLMs pueden mejorar su desempeño en problemas que requieren razonamiento aritmético, de sentido común o simbólico (Wei y cols., 2022).

La efectividad de estas técnicas depende tanto del modelo utilizado como de la calidad de la instrucción. El *Few-shot* permite tener un mayor control sobre la salida al incluir ejemplos explícitos, aunque es necesario seleccionarlos correctamente para evitar introducir sesgos. El COT añade transparencia y mejora el desempeño en tareas complejas, pero también genera respuestas más largas y más costosas.

El *prompting* es una forma simple de adaptar LLMs a contextos específicos sin necesidad de recurrir al *fine-tuning*, lo que reduce los costos de cómputo y aumenta la flexibilidad.

2.3.2. *Encoder-Only*: Bidirectional Encoder Representations from Transformers (BERT)

Los modelos *decoder-only* utilizan capas de atención causal, en las que la representación de cada *token* de entrada se ve determinada por los *tokens* que la preceden en la secuencia. En cambio, la arquitectura *Bidirectional Transformer Encoder*, utilizada en modelos como BERT, permite que la representación de cada *token* se calcule a partir de todos los *tokens*, incluidos los posteriores (Devlin y cols., 2019). Por lo tanto, los *embeddings* contextualizados generados para cada elemento de la entrada incorporan información de todo el texto. Esto resulta beneficioso para resolver tareas de etiquetado de secuencias, donde la

información contenida al final de una frase puede esclarecer o desambiguar el significado de palabras anteriores.

Los *embeddings* contextualizados obtenidos al procesar una secuencia con un modelo *Transformer* de tipo *encoder-only* pueden utilizarse como entrada de un modelo clasificador, por ejemplo, una red neuronal *feed-forward*.

Estos modelos suelen incorporar además *tokens* especiales. En particular, el *token* [CLS] se agrega al inicio de la secuencia de entrada y puede utilizarse para distintas tareas, como la clasificación de secuencias o la predicción del siguiente enunciado (*Next Sentence Prediction*). El vector de salida asociado a este *token* puede interpretarse como una representación condensada de la información del texto original. Por ello, se utiliza con frecuencia como entrada para capas posteriores encargadas de realizar tareas específicas, o como vector de representación global a partir del cual pueden calcularse medidas de similitud semántica entre textos.

En años posteriores a la publicación de BERT, se desarrollan variantes como «Robustly Optimized BERT Pre-training Approach» (ROBERTA) (Liu y cols., 2020) que mejoran los métodos de entrenamiento, logrando mejores resultados que los alcanzados por BERT. Posteriormente, se proponen arquitecturas alternativas como «Decoding-enhanced BERT with Disentangled Attention» (DEBERTA) (He, Liu, Gao, y Chen, 2021).

2.4. Multimodalidad

Los modelos de lenguaje basados en la arquitectura *Transformer* han sido adaptados para completar tareas *multimodales* caracterizadas por emplear simultáneamente diferentes formatos de información, como texto e imágenes. Una de las implementaciones más comunes en estos formatos es el *Vision Transformer* (VIT).

En esta solución, la imagen de entrada se divide en cuadrados uniformes, tratando cada parche como un *token* individual en una secuencia. Cada uno de estos parches se proyecta en un espacio vectorial mediante una capa lineal, lo que da como resultado un conjunto de *embeddings*. Adicionalmente, se incorpora una codificación posicional que permite al modelo aprovechar la estructura espacial original de la imagen.

Posteriormente, los vectores obtenidos son procesados a través de múltiples capas de autoatención, facilitando la captura de dependencias globales dentro de la imagen sin recurrir a capas convolucionales. Este enfoque obtuvo buenos resultados en comparación con modelos convolucionales del estado del arte en distintas tareas de clasificación de imágenes, especialmente al preentrenar al modelo con grandes cantidades de datos, además de requerir significativamente menos recursos computacionales para su entrenamiento (Dosovitskiy y cols., 2020). La Figura 2.3 muestra un esquema de esta arquitectura.

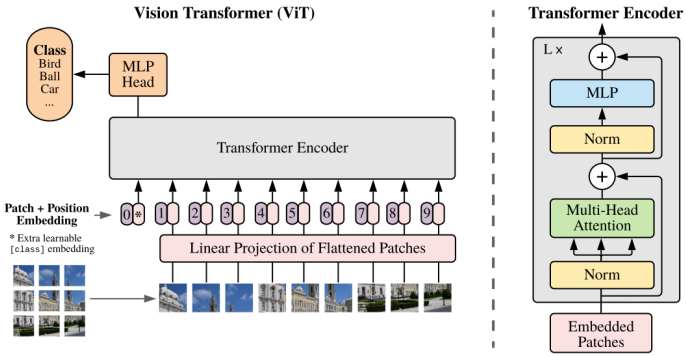


Figura 2.3: Diagrama de arquitectura *Vision Transformer*, utilizada para implementar algunos modelos multimodales (Dosovitskiy y cols., 2020).

2.5. Ajuste y optimización de grandes modelos de lenguaje

El entrenamiento de LLMs comienza con una etapa de preentrenamiento, en la que la red se expone a grandes volúmenes de texto para aprender representaciones generales del lenguaje. Esta etapa se realiza mediante tareas autosupervisadas como la predicción del siguiente *token* o el enmascarado de palabras. Un aspecto relevante del preentrenamiento es que es computacionalmente costoso debido a la cantidad de datos y parámetros involucrados.

Dado que repetir este proceso desde cero resulta inviable para las tareas del proyecto, se profundizará en formas de ajuste y optimización que permiten reutilizar modelos preentrenados y adaptarlos a tareas específicas.

2.5.1. Ajuste fino

El proceso de ajuste fino (*finetuning*) es una forma de aprendizaje por transferencia (*transfer learning*) que consiste en continuar el entrenamiento de una red neuronal preentrenada para adaptar su funcionamiento a un conjunto de datos específico.

Este enfoque busca aprovechar las capacidades de los modelos preentrenados para resolver problemas generales y, a partir de ellos, generar modelos adaptados a problemas específicos en los que se dispone de conjuntos de datos reducidos (Jurafsky y Martin, 2025).

El *finetuning* implica un costo computacional considerable, tanto en el número de operaciones como en la memoria requerida, especialmente cuando se aplica a modelos con un gran número de parámetros que deben ajustarse en cada paso

del entrenamiento. Incluso en modelos más pequeños, como los basados en la arquitectura BERT, el número de parámetros supera los cien millones, mientras que en modelos de mayor tamaño, como los de la familia LLAMA 3, esta cifra puede superar los tres mil millones.

Por esta razón, es común limitar el conjunto de parámetros a actualizar a un subconjunto, generalmente las últimas capas del modelo, mientras que el resto de los pesos permanece sin cambios. De esta forma, se reduce significativamente el número de cálculos necesarios, lo que disminuye el costo computacional total del proceso.

2.5.2. Optimización de hiperparámetros con *Optuna*

El proceso de aprendizaje se determina por una serie de parámetros externos al modelo, denominados hiperparámetros. Algunos ejemplos son la tasa de aprendizaje (*learning rate*), que controla el tamaño del paso en la optimización de la función de pérdida; el tamaño de lote (*batch size*), que determina cuántas muestras se procesan antes de actualizar los parámetros; y el número de épocas (*epochs*), que indica cuántas veces se recorre el conjunto de datos. Ajustarlos con precisión puede determinar en gran medida la calidad de los resultados.

Optuna es una herramienta que automatiza la búsqueda de estos hiperparámetros mediante la ejecución de múltiples pruebas (denominadas *trials*). Cada una corresponde a una configuración específica propuesta por la herramienta. Inicialmente, se define una función objetivo, la cual entrena y evalúa el modelo para cada conjunto de hiperparámetros sugerido. La métrica de rendimiento obtenida se utiliza para orientar la búsqueda hacia las configuraciones más prometedoras.

El método de optimización por defecto utilizado por *Optuna* es el *Tree-structured Parzen Estimator* (TPE), que es una variante de métodos clásicos de optimización *bayesiana*. La optimización *bayesiana* consiste en encontrar los hiperparámetros que maximizan o minimizan una función objetivo, en la que se busca un equilibrio entre la explotación y la exploración. La explotación implica buscar cerca de configuraciones que previamente produjeron buenos resultados, mientras que la exploración se enfoca en regiones no evaluadas del espacio de búsqueda. TPE, en particular, construye dos modelos de probabilidad: uno para las configuraciones de mejor desempeño y otro para las menos exitosas. Después de una fase inicial de exploración, *Optuna* ajusta la estrategia y selecciona los siguientes hiperparámetros priorizando los que maximizan la probabilidad de pertenecer al grupo exitoso y minimizan la probabilidad de estar entre los menos efectivos (Watanabe, 2023).

2.6. Reconocimiento óptico de caracteres

El reconocimiento óptico de caracteres (OCR) es la tarea de convertir el texto contenido en una imagen (ya sea mecanografiado, escrito a mano o impreso) en texto codificado que puede ser leído y procesado por una máquina.

Los primeros sistemas de OCR, patentados por Tausheck y Handel en la década de 1930, empleaban un método mecánico de emparejamiento de plantillas: la luz se proyectaba a través de máscaras de caracteres hacia un fotodetector para su identificación (Mori, Suen, y Yamamoto, 1992). Un avance posterior fue el sistema *Gismo* (observar la Figura 2.4) de M. Sheppard (1951) capaz de reconocer hasta 23 caracteres mecanografiados y traducirlos en señales digitales (Schantz, 1982). Posteriormente, el IBM 1418 de IBM (1960) fue una de las primeras máquinas en ser denominadas oficialmente «*Optical Character Reader*» (IBM, 1987).

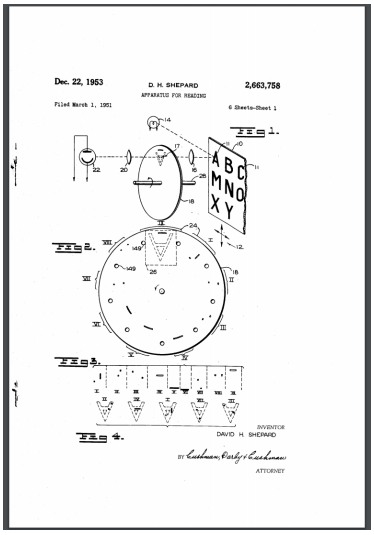


Figura 2.4: Machine Gismo OCR (Shugart, Alan, 1971)

Con el tiempo, el OCR incorporó técnicas de inteligencia artificial: se emplean modelos estadísticos, redes neuronales *feedforward*, así como convolucionales y recurrentes. Herramientas como *Tesseract* (Smith, 2013) permitieron reconocer texto en condiciones complejas, posibilitando que el OCR sea utilizado de forma masiva en procesos de digitalización.

Los modelos OCR se emplean ampliamente en procesos de digitalización. En particular, su aplicación a la documentación histórica facilita su preservación y accesibilidad, además de habilitar técnicas de extracción de información automática, como el reconocimiento de entidades nombradas.

En este contexto, estudios como «Assessing the Impact of OCR Quality on Downstream NLP Tasks» (Van Strien y cols., 2020) analizan cómo la precisión del OCR afecta tareas posteriores de procesamiento del lenguaje natural, demostrando que mejoras en la calidad del texto reconocido pueden aumentar significativamente el rendimiento en tareas posteriores como la extracción de entidades nombradas o la segmentación de enunciados.

2.6.1. Métodos convencionales de OCR

En general, el proceso de OCR se divide en varias fases. El artículo «A Survey on Optical Character Recognition System»(Islam, Islam, y Noor, 2017) identifica cinco subproblemas principales:

1. **Obtención de imágenes**: Consiste en el uso de herramientas externas, como cámaras y escáneres, para generar imágenes que contengan el texto a extraer.
2. **Preprocesamiento**: Se emplean técnicas de manipulación de imágenes para mejorar su calidad, como la reducción de ruido, *thresholding*, la binarización, la ecualización de histograma y la corrección de inclinación.
3. **Segmentación de caracteres**: Los caracteres presentes en la imagen son identificados y separados utilizando técnicas como el análisis de componentes conexas y la delimitación de líneas y palabras.
4. **Extracción de características**: Se obtienen atributos discriminativos de cada carácter. Por ejemplo: contornos, esquinas, histogramas de proyecciones, momentos invariantes, transformada de *Fourier* o mapas de características aprendidas por redes convolucionales.
5. **Clasificación de caracteres**: Cada conjunto de características se clasifica mediante algoritmos como *k-nearest neighbors* (K-NN), redes neuronales *feedforward* (MLP), redes neuronales convolucionales (CNN) o modelos recurrentes (por ejemplo, basados en *Long short term memory* (LSTM)), asignando a cada carácter su etiqueta correspondiente.

Algunos sistemas de OCR más sofisticados realizan variaciones sobre este esquema inicial. Por ejemplo, el modelo de código abierto *Tesseract*, desarrollado por HP entre 1985 y 1995 y actualmente mantenido por *Google*. Incluye en su arquitectura otros componentes, como el análisis de *layout* (disposición de elementos del documento) y múltiples instancias de reconocimiento de palabras.

Además de la segmentación de bloques, líneas y palabras mediante análisis de componentes conectados y proyecciones, *Tesseract* integra un motor basado en una red LSTM[1] para el reconocimiento de caracteres; un clasificador adaptativo de palabras que ajusta probabilidades según el contexto lingüístico y un sistema de corrección ortográfica con diccionarios multilingües.

Finalmente, incorpora un módulo de posprocesamiento que aplica reglas de gramática y puntuación para refinar los resultados (Smith, 2013). La figura 2.5 ilustra el procedimiento seguido por este modelo.

[1] Esta arquitectura está diseñada para facilitar la retención de información en secuencias largas.

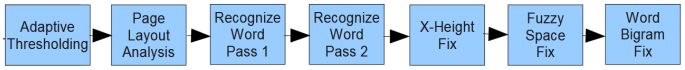


Figura 2.5: Diagrama de bloques de la arquitectura de *Tesseract* (Smith, 2013)

2.6.2. OCR a partir de LLMs multimodales

En los últimos años, los LLMs han comenzado a aplicarse de manera creciente en el área de OCR, incluida su aplicación para la transcripción de documentos históricos (Kim, Baudru, Ryckbosch, Bersini, y Ginis, 2025). A diferencia de las formas tradicionales, basadas en arquitecturas convolucionales o recurrentes, los LLMs multimodales (Sección 2.4) integran información visual y textual, lo que les permite abordar la tarea OCR como un problema de comprensión semántica.

Estos modelos reciben como entrada la imagen del documento y producen como salida la transcripción textual, aprovechando su capacidad de apoyarse en el contexto para desambiguar palabras que, sin información adicional, son difíciles de transcribir. Este enfoque permite aprovechar los conocimientos específicos del lenguaje y de distintos dominios aprendidos durante el entrenamiento en grandes conjuntos de datos.

2.6.3. Evaluación de Transcripciones

Al evaluar la calidad de una transcripción realizada por un sistema de OCR, se busca determinar el nivel de similitud entre el texto real del documento, usualmente denominado objetivo (*target*) o referencia, y el texto generado por el algoritmo de transcripción. Para esto, se utilizan medidas de distancia entre cadenas de caracteres, que pueden definirse en distintos niveles de granularidad, principalmente en carácter o palabra.

Distancia de *Levenshtein*

La distancia de *Levenshtein* (Levenshtein, 1966) permite calcular la distancia entre dos tiras de texto cuantificando el menor número de operaciones (inserciones, eliminaciones y sustituciones) necesarias para transformar una tira en otra. Esta medida de distancia se utiliza como base para varias métricas de similitud entre textos, entre ellas CER y WER (Rice, 1996).

Proporción de error en caracteres (CER)

La proporción de error en caracteres (*Character Error Rate*) normaliza la distancia de *Levenshtein* en función del largo total del texto objetivo. Esta métrica mide la proporción de caracteres erróneos al comparar el texto objetivo con la transcripción. Formalmente:

$$CER = \frac{S_c + D_c + I_c}{N_c} \tag{2.2}$$

Donde S_c representa el número total de sustituciones, D_c el número total de eliminaciones, I_c el número total de inserciones y N_c el número total de caracteres del texto objetivo. (Neudecker y cols., 2021)

Proporción de error en palabras (WER)

La proporción de error en palabras (*Word Error Rate*) extiende la formulación realizada en la distancia de *Levenshtein* a nivel de palabras. Su definición es análoga a la utilizada para CER:

$$WER = \frac{S_w + D_e + I_w}{N_w} \tag{2.3}$$

Donde S_w representa el número total de sustituciones de palabras, D_w el número total de eliminaciones de palabras, I_w el número total de inserciones de palabras y N_w el número total de palabras en el texto objetivo.

BLEU *Score*

El BLEU *Score* (Papineni, Roukos, Ward, y Zhu, 2002) es una métrica de evaluación comúnmente utilizada para evaluar traducciones automáticas. Esta se calcula a nivel de frases y se basa en la precisión de n-gramas. BLEU mide cuántos grupos consecutivos de palabras (n-gramas) en el texto generado[2] coinciden con los de las referencias. Además, incorpora una penalización por brevedad para evitar que traducciones excesivamente cortas obtengan puntuaciones altas.

El *BLEU Score* se define como:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.4}$$

donde:

- p_n es la precisión modificada para n-gramas, definida de forma simplificada como:
$$p_n = \frac{n\text{-gramas coincidientes}}{n\text{-gramas candidatos}} \tag{2.5}$$
- w_n es el peso asignado a cada n-grama (comúnmente $w_n = \frac{1}{N}$).
- BP es la penalización por brevedad, definida a continuación.

La penalización por brevedad se expresa mediante:

$$\text{BP} = \begin{cases} 1, & \text{si } c > r, \\ e^{\left(1-\frac{r}{c}\right)}, & \text{si } c \leq r, \end{cases} \tag{2.6}$$

[2] En el caso de su aplicación a OCR, la transcripción generada.

Donde c es la longitud de la traducción candidata y r es la longitud de la traducción de referencia.

Evaluación automática por coincidencia de caracteres (CHRF)

CHRF (Popović, 2015) es una métrica de evaluación derivada del *F-Score* y aplicada sobre n-gramas de caracteres. Por lo tanto, esta métrica calcula un agregado a partir de la evaluación tanto de la precisión (que es la proporción de n-gramas generados que aparecen en la referencia) como del *recall* (proporción de n-gramas de la referencia que aparecen en el texto generado).

Formalmente, puede ser definida como:

$$\text{chrF}_\beta = \frac{(1+\beta^2) \times \text{chrP} \times \text{chrR}}{\beta^2 \times \text{chrP} + \text{chrR}} \tag{2.7}$$

La métrica CHRF combina CHRP (precisión) y CHRR (*recall*) mediante un parámetro de ponderación β, donde:

- **CHRP**: Porcentaje de n-gramas de caracteres presentes en la hipótesis (en este caso, la transcripción realizada por el sistema) que también ocurren en la referencia.
- **CHRR**: Porcentaje de n-gramas de caracteres presentes en la referencia que también ocurren en la hipótesis.

2.7. Reconocimiento de entidades nombradas

La extracción de información es una subdisciplina del PLN que busca automatizar el proceso de obtener información estructurada a partir de textos.

Convencionalmente, el problema se divide en varias tareas, cada una de ellas dirigida a extraer componentes específicos de la semántica del texto. Entre ellas se encuentran el reconocimiento de entidades nombradas (NER), la extracción de relaciones y de eventos, entre otras. A partir de su resolución, es posible elaborar modelos de datos, como grafos de conocimiento, que facilitan la consulta de información (Jurafsky y Martin, 2025).

En este contexto, el reconocimiento de entidades nombradas consiste en identificar segmentos de texto que referencian entidades particulares y clasificarlos adecuadamente.

Las categorías objetivo son predefinidas y dependen del problema a resolver. Usualmente, estas incluyen personas, lugares y organizaciones (Jurafsky y Martin, 2025) (Keraghel, Morbieu, y Nadif, 2024).

La Figura 2.6 es un ejemplo de texto con sus entidades etiquetadas. En este caso existen dos entidades de tipo *persona* (Winston y Julia), ambas compuestas de una sola palabra, y una de tipo *organización*, compuesta de dos palabras.

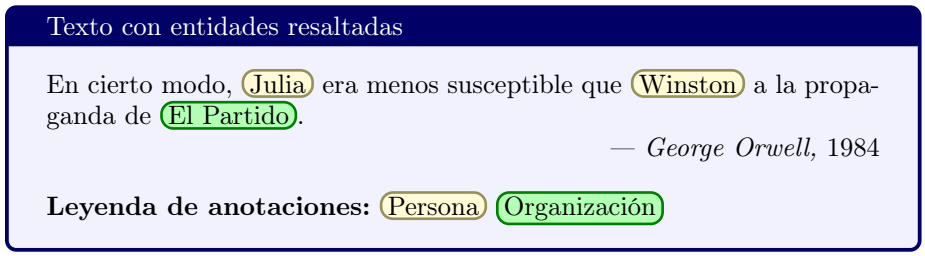


Figura 2.6: Ejemplo de texto con entidades etiquetadas.

Formalmente, el reconocimiento de entidades nombradas puede formularse como un problema de etiquetado de secuencias. Dada una secuencia $T = \{t_1, t_2, \ldots, t_n\}$ de n *tokens*, la tarea consiste en generar un conjunto de tuplas (I_s, I_e, ℓ). En cada tupla, I_s indica el índice de inicio de la entidad y I_e el de fin, siendo clasificada con la etiqueta ℓ (Keraghel y cols., 2024).

En la práctica, existen múltiples formas de representar el etiquetado de secciones del texto correspondientes a entidades.

Etiquetado BIO

Una forma común de etiquetado es el formato *Begin-Inside-Outside* (BIO). Este método asigna a cada palabra o subpalabra una etiqueta. Por ejemplo, para detectar entidades de tipo persona se utiliza «B-PER» para la primera palabra de la entidad, «I-PER» para todas las siguientes dentro de esa entidad y, finalmente, «O» cuando no corresponde a ninguna entidad. Una desventaja de este tipo de etiquetado es que no permite representar entidades anidadas; por ejemplo, como se ilustra en la Figura 2.7, el «*Ministerio de Defensa de Uruguay*». El esquema obliga a elegir entre etiquetar la frase completa como organización o marcar «*Ministerio de Defensa*» como organización y «*Uruguay*» como lugar.

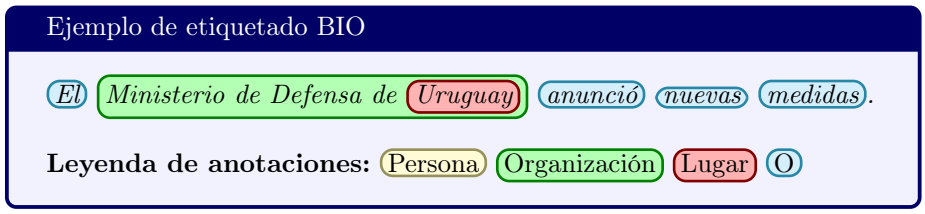


Figura 2.7: Ejemplo de texto con entidades anotadas.

2.7.1. Métodos para NER

Las técnicas utilizadas para el reconocimiento de entidades nombradas han progresado significativamente desde su origen en 1996.

Al igual que en muchas otras tareas de PLN, los primeros sistemas de NER utilizaban métodos basados en reglas definidas manualmente y en información lexicográfica. Posteriormente, se introduce el uso de aprendizaje automático basado en ingeniería de características, principalmente utilizando métodos de aprendizaje supervisado. A partir del año 2011, prolifera la aplicación de técnicas de aprendizaje profundo, cuya capacidad de capturar patrones complejos logra mejorar las capacidades de los sistemas de NER disponibles. Entre ellas se encuentran las redes convolucionales (CNN) y redes recurrentes como las basadas en capas LSTM o *Gated Recurrent Unit* (GRU) (Munnangi, 2024) (Keraghel y cols., 2024).

La Figura 2.8 presenta un diagrama de la taxonomía de métodos de NER.

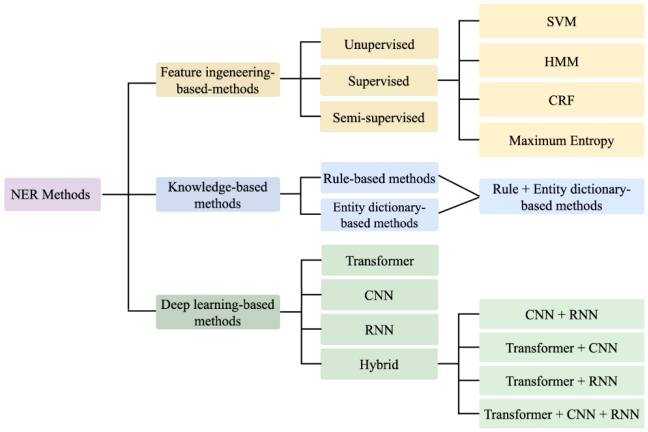


Figura 2.8: Principales métodos para NER. Figura extraída de (Keraghel y cols., 2024)

Métodos basados en conocimiento

Los métodos basados en conocimiento buscan explotar el conocimiento del dominio para identificar entidades en el texto. Se emplean colecciones de entidades correspondientes a un dominio particular, a veces llamadas *gazetteers*, para detectar entidades, por ejemplo, nombres propios presentes en datos externos. Los *gazetteers* son colecciones estructuradas que incluyen nombres de personas, lugares, organizaciones y otros términos específicos, lo que facilita la identificación rápida de entidades conocidas. Asimismo, los lexicones (lista de términos pertenecientes al vocabulario de un dominio específico) amplían el vocabulario al incluir términos técnicos, jerga y variaciones lingüísticas propias del dominio, lo que mejora la cobertura y precisión en la extracción de información (Keraghel y cols., 2024).

Adicionalmente, se utilizan reglas predefinidas, basadas en gramáticas o en criterios sintácticos, para identificar entidades en el texto. Estas reglas, a veces

elaboradas manualmente, permiten reconocer entidades incluso cuando no están registradas en los diccionarios utilizados. Por ejemplo, el prefijo *«Mr.»* puede utilizarse para identificar entidades de tipo *Persona*, aunque su nombre no esté incluido en los datos (Sekine y Nobata, 2004).

2.7.2. Modelos basados en *transformer*

Para la tarea NER suelen utilizarse modelos de tipo BERT (arquitectura *decoder-only*), asignando a cada *token* una etiqueta siguiendo el esquema BIO.

Dado que las palabras que forman la secuencia son *tokenizadas*, es necesario determinar las etiquetas correspondientes a cada *token* a partir de las asignadas a cada palabra en los datos de entrenamiento. Asimismo, al momento de la inferencia, se debe asignar a cada palabra una etiqueta derivada de las predicciones de los *tokens* que la conforman.

Los modelos basados en *Transformer* han demostrado un rendimiento superior al de arquitecturas tradicionales para la tarea de reconocimiento de entidades nombradas, como *BiLSTM-CRF*, en conjuntos de datos de dominio general como CONLL-2003. Sin embargo, en conjuntos correspondientes a dominios específicos, como BIONLP2004 (biomedicina), las arquitecturas convencionales continúan mostrando capacidades similares o incluso mayores (Keraghel y cols., 2024).

2.7.3. Evaluación de etiquetado en NER

Es posible interpretar el problema de reconocimiento de entidades nombradas como una tarea de clasificación de palabras. Debido a esto, las principales métricas utilizadas para evaluar su rendimiento son las mismas que se emplean comúnmente en modelos de clasificación en general: *precisión*, *recall* (cobertura) y F_β , que se definen como:

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2.8}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}$$

$$F_\beta = (1 + \beta^2) \times \frac{\text{Precisión} \times \text{Recall}}{\beta^2 \times \text{Precisión} + \text{Recall}} \tag{2.10}$$

Donde:

- TP (*True Positives*) son las entidades correctamente identificadas.
- FP (*False Positives*) son las entidades etiquetadas incorrectamente.
- FN (*False Negatives*) son las entidades que el sistema no detectó.

Una precisión alta en una clase indica que el modelo comete pocos errores al clasificar palabras de esa clase. Es decir, la cantidad de falsos positivos es baja.

El *recall* (cobertura) mide qué proporción de las palabras de una clase el modelo identifica correctamente; es alto cuando hay pocos falsos negativos.

El F_β combina la precisión y el *recall* y su valor puede utilizarse para medir el rendimiento general del modelo. El parámetro β permite ajustar la importancia relativa entre la precisión y el *recall* según las necesidades específicas de la tarea de clasificación.

Cuando $\beta = 1$ (F_β), se otorga el mismo peso a la precisión y al *recall*. Si $\beta > 1$, se da mayor importancia al *recall*, lo cual resulta útil en contextos en los que es crítico minimizar los falsos negativos. Por el contrario, si $\beta < 1$, se prioriza la precisión, siendo apropiado cuando es más importante evitar falsos positivos.

Es común que la evaluación de modelos de clasificación multiclase utilice la métrica $F1_{\text{macro}}$, la cual se calcula como el promedio de las F1 de cada clase evaluada.

En el contexto NER, la evaluación se realiza a nivel de entidad, lo que introduce desafíos adicionales en comparación con tareas como la categorización de texto o el etiquetado de categorías gramaticales (*POS tagging*). Por ejemplo, si un modelo identifica «Juan» como entidad (PER) pero no «Juan Pedro», se generan dos errores: un falso positivo para «O» (se clasifica «Pedro» como fuera de la entidad) y un falso negativo para «I-PER» (no reconoce que «Pedro» pertenece a la entidad persona) (Jurafsky y Martin, 2025).

2.8. Generación de datos sintéticos

Los métodos de aprendizaje supervisado, especialmente las redes neuronales con un número alto de parámetros, requieren conjuntos de datos con una gran cantidad y variedad de elementos. La escasez de datos puede resultar en modelos que no generalicen adecuadamente lo aprendido durante el entrenamiento. Para enfrentar este problema, se emplean técnicas de aumento de datos (*data augmentation*). Por ejemplo, en el procesamiento de imágenes, es común aumentar la cantidad de datos mediante modificaciones a imágenes reales pertenecientes al conjunto de entrenamiento.

Alternativamente, es posible generar de forma automática nuevos datos etiquetados, denominados datos sintéticos. Los datos sintéticos no se generan a partir de modificaciones de elementos del conjunto de entrenamiento, sino que se emplean métodos basados en reglas o en modelos generativos para generar ejemplos nuevos.

Las técnicas de *data augmentation* han sido utilizadas con éxito en múltiples áreas del procesamiento del lenguaje natural, como el análisis de sentimiento. Sin embargo, los autores del estudio «A survey of data augmentation in named

entity recognition» (Huang, Gao, y Ren, 2025) afirman que se subutilizan en la tarea de NER en particular.

Una reciente revisión sistemática (Goyal y Mahmoud, 2024) define una clasificación de las estrategias de generación de datos sintéticos en varios formatos; a efectos de este análisis, se considerarán únicamente trabajos centrados en la generación de texto, por ser el formato pertinente al enfoque del presente proyecto.

El estudio organiza la generación de estos datos en dos categorías: la **generación programática** y el enfoque de **aprendizaje automático**.

2.8.1. Enfoque Programático

En esta categoría, los datos se generan mediante la aplicación de reglas, construcciones lógicas y modelos de simulación para que se asemejen a las cualidades y comportamientos del mundo real.

En el ámbito del PLN, las técnicas basadas en reglas suelen implementarse mediante gramáticas formales, como las gramáticas libres de contexto o las gramáticas probabilísticas. Estos sistemas permiten crear datos con un alto grado de control sobre la estructura de los textos generados.

El trabajo «*Grammar-based Data Augmentation for Low-Resource Languages: The Case of Guarani-Spanish Neural Machine Translation*» (Lucas y cols., 2024) es un ejemplo de la efectividad de los sistemas de reglas basados en gramáticas para la generación de datos sintéticos, en este caso, aplicados al entrenamiento de un modelo de traducción automática en un contexto de escasez de datos.

2.8.2. Técnicas basadas en aprendizaje automático

Las técnicas de aprendizaje automático para la generación de datos sintéticos buscan modelar la distribución de los datos reales. Entre los métodos utilizados se encuentran los autoencoders variacionales (VAE), las redes generativas antagónicas (GAN) y, para datos textuales, LLM.

Los LLMs pueden ser utilizados para generar datos sintéticos aprovechando su capacidad para interpretar y sintetizar texto similar al humano a partir de grandes volúmenes de datos de entrenamiento. Estos modelos permiten generar ejemplos guiados por instrucciones o ejemplos (ver Sección 2.3.1) y producir salidas con estructuras determinadas. Para mejorar la controlabilidad, se puede complementar con restricciones en la decodificación a través de la temperatura, *top-p sampling*, entre otros (ver Sección 2.3.1). Con este tipo de técnicas, es posible generar ejemplos coherentes con el dominio sin agregar un costo de anotación manual.

2.8.3. Entrenamiento utilizando datos sintéticos

Para generar datos sintéticos en el contexto de NER, es necesario tanto generar el contenido como sus etiquetas. Los ejemplos pueden ser generados por LLMs siguiendo un formato de anotación compatible con el conjunto de entrenamiento real. En (Dao, Teranishi, Matsumoto, y Aizawa, 2025) se propone una generación basada en entidades para el dominio biomédico, para luego construir un *prompt* que exige incluirlas con su anotación. Una vez obtenido un conjunto de datos sintéticos, se identifican dos enfoques para incorporar el lote a los datos de entrenamiento:

- **Entrenamiento Integrado (IAT)**: Del inglés *Integrated Augmented Training*, los datos sintéticos se incorporan al conjunto de datos original, expandiendo el conjunto de entrenamiento. De esta manera, el modelo simultáneamente muestra ejemplos reales y ejemplos generados artificialmente, lo que permite aumentar la diversidad de los ejemplos y reducir el riesgo de sobreajuste en un conjunto de datos limitado. Sin embargo, este método puede introducir ruido si la calidad de los datos sintéticos no es buena con respecto a los datos originales, lo que podría degradar el rendimiento final.
- **Entrenamiento Secuencial (SAT)**: Del inglés *Sequential Augmented Training*, consiste en una primera instancia de entrenamiento únicamente con datos sintéticos, para luego ser entrenado con el conjunto de datos original. La motivación principal es que el entrenamiento inicial con ejemplos sintéticos actúe como una fase de preentrenamiento, permitiendo al modelo adquirir patrones generales, mientras que la segunda etapa con datos reales ajusta y corrige esas representaciones hacia la distribución verdadera.

El estudio citado utiliza los modelos BERT y PUBMEDBERT sobre dos conjuntos de datos en el dominio biomédico. Se concluye que en casos de escasez de datos reales, IAT supera a SAT, con menos de 200 ejemplos de datos reales en uno o menos de 50 en el otro. Para casos entre 200 y 2000 ejemplos, SAT supera a IAT y al entrenamiento solo con datos reales; el preentrenamiento con datos sintéticos captura patrones generales y el ajuste con datos reales corrige el sesgo y el ruido. Con más de 2000 ejemplos, SAT supera consistentemente a IAT.

Capítulo 3

Recopilación de antecedentes locales

El *Instituto de Computación de la Facultad de Ingeniería* (INCO-FING) participa en el procesamiento de documentos a través de los grupos de *Gestión de Datos y Modelado* (GEMA) y de *Procesamiento del Lenguaje Natural* (PLN). En este contexto, se han realizado múltiples aportes que sirven como antecedentes relevantes para este proyecto.

La descripción general del proyecto CRUZAR se presenta en una publicación (Etcheverry y cols., 2021), que recopila los avances logrados hasta el año 2021 y presenta las líneas de trabajo desarrolladas posteriormente.

3.1. *Pipeline* de IE

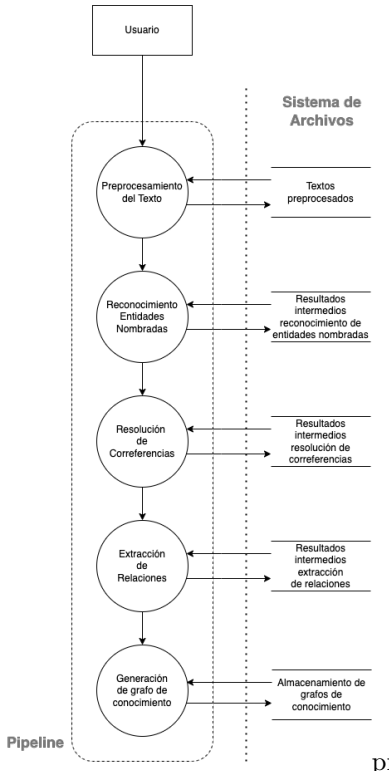
La extracción de información es central en el desarrollo de CRUZAR. De esta manera, se han desarrollado esfuerzos para modelar procesos que permitan obtener información estructurada a partir de las transcripciones.

En particular, el proyecto de grado (?, ?) implementa un *pipeline* de extracción de información que incluye el reconocimiento de entidades nombradas, la resolución de correferencias, la extracción de relaciones y la generación de un grafo de conocimiento.

Se realizó una tarea de anotación de un corpus utilizando la herramienta INCEPTION, que permite corregir y validar las anotaciones de forma colaborativa. El corpus anotado generado a partir de este proceso permite mejorar los módulos de extracción de información.

El *pipeline* parte de las transcripciones y termina con la persistencia en un modelo de datos. Este flujo está compuesto por cinco etapas, como se muestra

en la Figura 3.1.



proyecto

Figura 3.1: Esquema inicial del *pipeline* propuesto por (Cardozo Ramírez y cols., 2021)

proyecto

El *pipeline* completo incluye pasos adicionales que no están contemplados en la Figura 3.1. Mientras que el proyecto de grado de Cardozo y cols. aporta a la parte central del proceso, centrada en la extracción de información y la generación del grafo, el *pipeline* general abarca también etapas previas y posteriores.

En el *pipeline* completo, en primer lugar, se realiza la transcripción de las imágenes y la corrección de las transcripciones y, a partir de ellas, se ejecuta el proceso de extracción de información, que incluye el reconocimiento de entidades nombradas, la resolución de correferencias y la extracción de relaciones.

Para la implementación del módulo NER, se evalúan distintas herramientas de PLN y un enfoque basado en patrones definidos manualmente. A partir de los resultados obtenidos en este proceso, se genera un grafo *Resource Description Framework* (RDF) por cada documento. Posteriormente, las salidas de cada documento se unifican en un único resultado. Este paso requiere resolver la tarea de coincidencia de entidades (*Entity Matching*).

Otro antecedente relacionado es la plataforma LUZ, donde se busca construir una base de conocimiento que integre la información extraída a partir de las transcripciones. Para esto, se busca utilizar estándares propios de la Web Semántica, como RDF y *Web Ontology Language* (OWL). En este proyecto se utiliza una ontología que define los conceptos principales del dominio, las relaciones que estos establecen y las propiedades de dichas relaciones.

Dado que el resultado de procesar cada documento es un grafo que sigue un esquema compartido, se debe incluir un paso de integración semántica (también llamado *Entity Matching* o *Semantic Reconciliation*), que consiste en identificar y unificar nodos que se refieren a la misma entidad, aunque aparezcan con nombres o formatos distintos. Para ello, se exploran soluciones basadas en reglas.

3.2. Antecedentes OCR

En años recientes, se han realizado múltiples proyectos y tesis que han estudiado la aplicación de herramientas de OCR sobre estos documentos.

En 2020, se desarrolla el proyecto «Procesamiento de Lenguaje Natural para la reconstrucción de textos a partir de imágenes correspondientes a archivos históricos de la década del 70», que se centra en la reconstrucción de textos a partir de imágenes. El trabajo realiza un análisis comparativo de distintas herramientas de OCR (*Tesseract 4*, *ABBYY FineReader 14* y *Readiris 17*) para la generación de transcripciones (Stabile, Fernández, y Fioritto, 2020a).

Las transcripciones generadas son, posteriormente, procesadas con el fin de obtener resultados más fiables. El trabajo propone y evalúa el uso de técnicas de PLN basadas en modelos de lenguaje tradicionales y en traducción estadística automática (Figura 3.2).

Se realiza un preprocesamiento destinado a normalizar algunos aspectos de los textos obtenidos, empleando el reemplazo de caracteres. Esto incluye la corrección de errores comunes y otras normalizaciones de los caracteres especiales contenidos en el texto, como comillas o *ampersand*. Adicionalmente, se implementa un proceso que utiliza diccionarios para identificar y unificar palabras que fueron divididas erróneamente.

El uso de modelos de lenguaje basados en n-gramas para la corrección de transcripciones sigue tres pasos principales: detección de errores, obtención de posibles correcciones y selección de la corrección más adecuada.

Otro enfoque evaluado en este proyecto es el uso de modelos de traducción automática estadística. En este caso, se conceptualiza el problema como la traducción de un texto incorrecto generado mediante OCR a un texto correcto. Para esto, se construye un corpus paralelo, conformado por un conjunto de textos con errores y sus traducciones.

La corrección de transcripciones basada en modelos de lenguaje de n-gramas

logró eliminar un número significativo de errores. Aproximadamente 71 % de los documentos evaluados obtuvieron un aumento en su puntaje de similitud. Por otro lado, la solución basada en traducción automática no fue efectiva; en este caso, aproximadamente 75 % de los ejemplos sufrieron una disminución de su puntaje.

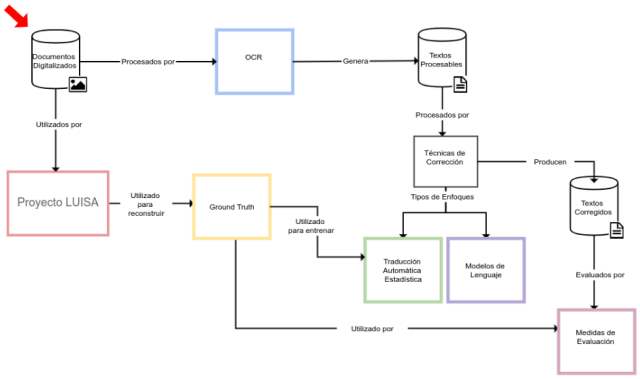


Figura 3.2: Esquema general del proceso de transcripción y evaluación del proyecto de reconstrucción de textos (Stabile y cols., 2020a).

Un enfoque posterior (Etcheverry y cols., 2021) propuso un proceso híbrido, donde se realizó una transcripción inicial utilizando el motor de *Tesseract 4.1*, que luego fue corregida manualmente. Para determinar qué transcripciones requieren ajustes, se desarrolló una métrica de evaluación de legibilidad denominada *readability score*, diseñada para estimar la calidad de la transcripción cuando no se dispone del texto correcto contenido en la imagen original.

Este enfoque atiende la necesidad de incorporar nuevas estrategias ante el gran tamaño del corpus y la baja calidad de las imágenes, lo que hace que *«incluso los OCRs comerciales más sofisticados sean ineficaces para una parte significativa de los archivos»*.

El *readability score* (r) se calcula a partir de un *diccionario enriquecido* (D). Dado un texto transcrito $t = \{w_1, w_2, \ldots, w_n\}$, el *score* $r(t)$ se define como:

$$r(t) = \sum_{l=1}^{L} a_l p_l(t) \tag{3.1}$$

Donde $p_l(t)$ representa la proporción de palabras de largo l en t que aparecen en el diccionario D, y a_l es un coeficiente determinado a partir de datos transcritos manualmente. Se utiliza una regresión lineal para aproximar un conjunto de pesos $(a_1, \ldots, a_L)$ tal que, para cada documento j, su *readability score* $r(t_j)$ sea lo más parecido posible a un puntaje ideal $s(t_j)$, calculado a partir de la distancia de *Levenshtein* entre la transcripción manual y la obtenida por OCR.

Se considera que una transcripción es suficientemente buena cuando su *readability score* es mayor o igual a 60 %. Las transcripciones son posteriormente corregidas según los procedimientos definidos en el proyecto mencionado en la sección 3.2. Este antecedente es especialmente relevante para el desarrollo de esta etapa, ya que los puntajes son útiles para categorizar las imágenes transcritas según su calidad.

Posteriormente, en el proyecto de grado «*Modelos Seq2Seq para la transcripción de documentos del Archivo Berrutti*» (Chavat Pérez, 2022), se evalúan métodos de aprendizaje automático para la transcripción de documentos. En particular, se implementa una red de tipo *encoder-decoder*. El modelo toma como entrada segmentos de imagen correspondientes a cada documento y los procesa utilizando una red convolucional. En este proceso, se generan vectores de características que luego se utilizan para obtener las codificaciones finales mediante una red LSTM bidireccional de tres capas. Posteriormente, el decodificador utiliza una red LSTM unidireccional y una capa *feed-forward* para generar una distribución probabilística sobre los símbolos existentes.

Por otro lado, el artículo «*Building tools to analyze the files of the Uruguayan dictatorship: information extraction from the personal records of organización coordinadora de operaciones antisubversivas (OCOA)*», publicado en el año 2024, se centra en definir una metodología para la extracción de información de una clase particular de documentos: fichas estructuradas que contienen información personal generada por la OCOA (Nogueira, Etcheverry, y Randall, 2024a).

Las fichas mantienen un formato y dimensiones consistentes. En una misma imagen pueden incluirse las partes frontal y trasera, ya que las fichas originales tienen contenido impreso o mecanografiado en ambos lados. Cada ficha se compone de múltiples campos. El proceso de extracción divide e identifica las distintas partes y campos, y aplica otros ajustes (como la corrección de la orientación) para facilitar el proceso de transcripción y etiquetado de datos (Figura 3.3).

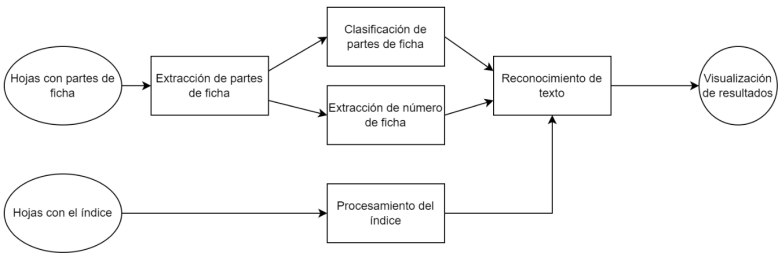


Figura 3.3: Diagrama con principales etapas de la metodología, imagen extraída de (Nogueira y cols., 2024b)

proyecto

El procedimiento utilizado para la identificación de partes de ficha en cada imagen calcula un vector de suma horizontal (S_h), que se conforma por la sumatoria de los valores de cada píxel de la fila. A partir de este vector, se determina qué rango de filas corresponde a cada parte.

Las partes de la ficha se clasifican posteriormente mediante un algoritmo de *template matching*. A partir de esta clasificación se obtiene también la ubicación de las distintas etiquetas que identifican los campos. Luego, la posición de la información contenida en cada campo se determina de forma relativa a la de la etiqueta.

Para realizar el reconocimiento de texto de las imágenes se utilizan los modelos de OCR *Calamari* y *Tesseract*. Se utiliza *finetuning* para mejorar los resultados, a partir de los datos etiquetados obtenidos de LUISA. Además, se busca corregir algunos resultados utilizando diccionarios; en particular, se genera un diccionario de organizaciones políticas, conformado por nueve siglas. Se corrigen las palabras cuya distancia de edición sea menor o igual a 1. Complementariamente, se utiliza un diccionario de apellidos de forma análoga.

Luego, en el proyecto «IA para el procesamiento de archivos documentales y su aplicación al caso de los archivos del pasado reciente» se avanzó en las tareas asociadas a la transcripción de imágenes del archivo a textos (Sastre, Etcheverry, Rey, Moncecchi, y Rosá, 2025), donde se plantea un método de poscorrección de la salida del OCR mediante el uso de LLMs combinados con una estrategia de decodificación restringida (Figura 3.4).

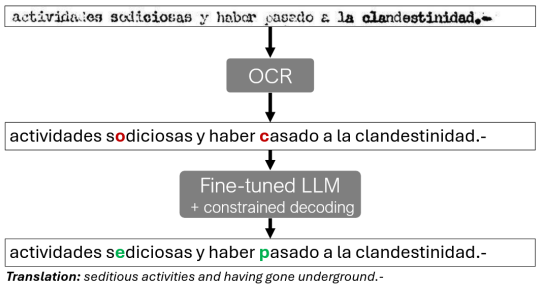


Figura 3.4: Proceso de transcripción y corrección de textos propuestos en el proyecto

Finalmente, en la tesis de maestría «*Teacher Student Curriculum Learning applied to Optical Character Recognition: An analysis based on a case study*» (Laguna Queirolo, 2025) se explora la aplicación de *Teacher Student Curriculum Learning* (TSCL) en la tarea de OCR, utilizando los documentos del Archivo Berrutti. Esta técnica mejora las métricas reportadas por Chavat en CER en un 16 %.

3.3. Antecedentes NER

Los avances locales en NER sirvieron como antecedentes relevantes para el desarrollo del presente proyecto de grado.

El proyecto de investigación «IA para el procesamiento de archivos documentales y su aplicación al caso de los archivos del pasado reciente»(Etcheverry, Rosá, Gómez, y Randall, s.f.) aborda problemas similares de aplicación de IA en el contexto de CRUZAR, en particular en la mejora de la transcripción y la extracción de información.

Asimismo, en el proyecto «LUZ: un sistema de búsqueda sobre los archivos de la última dictadura militar»(Cardozo Ramírez y cols., 2021), se evaluó el desempeño de múltiples herramientas genéricas y el uso de patrones y diccionarios. Esta metodología depende en gran medida de patrones predefinidos, diccionarios estáticos y herramientas que no se han ajustado a las particularidades del dominio. Los textos son extraídos de los *microfilms* a partir de técnicas de OCR, por lo que cualquier error en la extracción (como caracteres faltantes, errores de reconocimiento de letras similares o problemas de segmentación) puede impedir el reconocimiento de un patrón, incluso si se encuentra presente en el texto.

Los mejores resultados de la etapa NER de este último antecedente se toman como línea base en el Capítulo 6.

Otro antecedente de interés fue la tesis ya mencionada, en la que se procesaron fichas del OCOA (Nogueira y cols., 2024a), la cual abordó integralmente la extracción de entidades, relaciones y eventos, junto con sus argumentos, utilizando LLM entrenados para la generación de código.

Por otro lado, la tesis de maestría de Rodrigo Gallardo, defendida en agosto de 2025, también aborda la tarea de extracción de información y, en particular, la de NER. Dado que la tesis se desarrolló en paralelo a nuestro proyecto, no fue posible utilizarla como referencia comparativa.

Capítulo 4

Un *pipeline* de procesamiento de documentos históricos uruguayos

En este capítulo se presenta el problema a resolver y el diseño de un prototipo de *pipeline* para transcribir y extraer entidades nombradas (personas, organizaciones y lugares) a partir de imágenes de documentos históricos uruguayos deteriorados. El sistema se organiza en dos componentes principales: un módulo de reconocimiento óptico de caracteres y un módulo de reconocimiento de entidades nombradas, ambos implementados con modelos basados en *Transformer*.

El objetivo es desarrollar un prototipo de *pipeline* que permita evaluar e integrar distintos motores de OCR y modelos NER, adaptados a la complejidad y al deterioro de los documentos. Parte de su utilidad radica en la independencia de cada componente, lo que permite a los desarrolladores e investigadores abstraerse de las secciones del sistema ajenas a su trabajo y concentrarse exclusivamente en las que les corresponden.

Por ejemplo, un investigador interesado únicamente en la tarea de NER puede fijar el OCR a utilizar, o bien ignorar esta etapa si ya cuenta con transcripciones para experimentar.

4.1. *Pipeline* completo

Si bien el alcance del presente proyecto se limita a las etapas de OCR y NER, desde CRUZAR existen planes y esfuerzos por continuar el desarrollo de las etapas posteriores de IE. Tomando como referencia la Figura 3.1, se describen todas las etapas del *pipeline*. El libro *Speech and Language Processing* de (Jurafsky y Martin, 2025) profundiza en estas tareas y sirve de base para su definición.

Entrada de imágenes: Los documentos históricos escaneados son recibidos en formatos estándar de imagen (PNG o JPG). La conexión con la base de datos de LUZ no fue realizada por políticas del proyecto CRUZAR; para poder acceder a los datos es necesario realizar una solicitud a un administrador del sistema. Una vez obtenidos los documentos, se procede a transcribirlos.

Módulo OCR: Los documentos, luego de ser escaneados, son procesados por un motor OCR que obtiene transcripciones en texto plano a partir de la imagen. Esta etapa debe adaptarse a las particularidades de los documentos deteriorados del dominio.

Módulo NER: Luego de obtener las transcripciones, el modelo NER procesa el texto e identifica las entidades nombradas, clasificándolas según su tipo.

Resolución de correferencias: El objetivo de la correferencia es identificar cuándo distintas menciones se refieren a la misma entidad. Esta etapa se encarga de consolidar una entidad por documento o por colección de documentos y asignar a cada mención su identificador. Un ejemplo de correferencia se muestra en el siguiente texto:

Extracción de relaciones

Winston se mantuvo de espaldas a la telepantalla... A un kilómetro de su casa, el Ministerio de la Verdad, donde ϵ[a] era empleado, elevaba su inmensa y blanca mole sobre un panorama de tintes sombríos. — *George Orwell,* 1984 **Entidades canónicas:** Winston

Ministerio de la Verdad

[a]Esta correferencia es conocida como sujeto omitido o tácito.

Extracción de relaciones: La siguiente tarea, luego de la extracción de entidades y la resolución de correferencias, es la de comprender cómo ellas se relacionan entre sí. Esta es una tarea central en el área del PLN, y se puede resumir como los esfuerzos por responder la pregunta *«¿Quién le hizo qué a quién?»*

Un ejemplo de extracción de relaciones puede ser el siguiente texto:

Extracción de relaciones

Winston Smith trabaja en el Ministerio de la Verdad, donde se encarga de reescribir artículos para alinearlos con los discursos del Gran Hermano.

— *George Orwell,* 1984

Leyenda de anotaciones: Persona Organización
Relaciones presentes:

- (`Winston Smith`, *trabaja en*, `Ministerio de la Verdad`)
- (`Winston Smith`, *reescribe para*, `Gran Hermano`)

Extracción de Eventos: La extracción de eventos busca reconocer en el texto situaciones junto con su tipo y sus participantes. A diferencia de las relaciones, que conectan dos entidades, un mismo evento puede involucrar múltiples participantes con atributos espaciales y temporales.

Extracción de eventos

Winston Smith trabaja en el Ministerio de la Verdad, donde se encarga de reescribir artículos para alinearlos con los discursos del Gran Hermano.

— *George Orwell,* 1984

Eventos identificados:

- **Ejemplo de evento:** *encargarse de una tarea*
  - **Agente:** Winston Smith
  - **Tarea:** reescribir artículos
  - **Contexto:** dentro de su rol en el Ministerio de la Verdad

Persistencia en un modelo de datos de grafo: La extracción de relaciones y eventos es fundamental para estructurar la información contenida en grandes volúmenes de texto, ya que permite integrarla en grafos de conocimiento. Estas representaciones facilitan la organización, consulta y análisis de la información, al modelar de manera explícita las conexiones y dependencias identificadas en los textos mediante un modelo de datos basado en grafos.

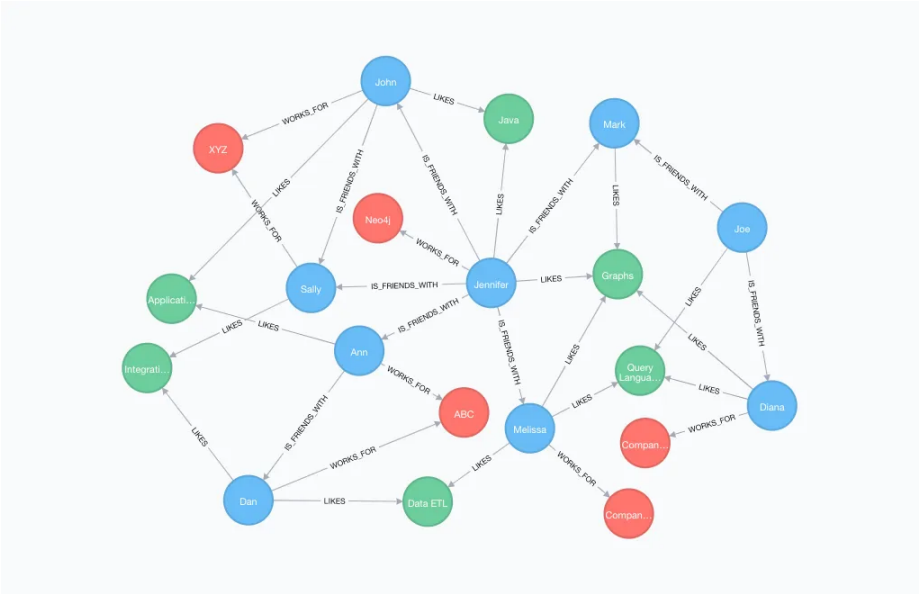


Figura 4.1: Ejemplo de visualización de una base de datos de grafo, los nodos representan a las entidades y las aristas a las relaciones entre ellas [1].

[1]Base de datos de grafo: https://softwareengineeringwk.substack.com/p/nosql-sql-graph-database-types

4.2. Etapas desarrolladas en el proyecto

A continuación se describe el prototipo implementado, centrado en las etapas OCR y NER. El objetivo es desarrollar un prototipo que permita evaluar e integrar distintos motores de OCR y modelos NER, adaptados a la complejidad y al deterioro de los documentos. La Figura 4.2 muestra el flujo general: la imagen es escaneada, luego es procesada por un módulo OCR que genera la transcripción en texto plano y, por último, sus entidades son etiquetadas en formato BIO por el modelo NER.

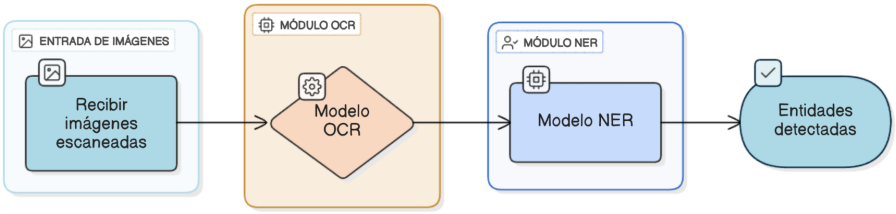


Figura 4.2: Diagrama de flujo del prototipo

4.2.1. Módulo OCR

El *pipeline* permite elegir entre diferentes motores tradicionales de OCR, como *Tesseract* o *MonkeyOCR*, y modelos de lenguaje multimodales, como LLAMA o MINICPM.

Cada uno de estos modelos se gestiona dinámicamente mediante un cargador interno que permite ajustar el flujo de procesamiento según el motor seleccionado.

Para los modelos basados en LLMs, se diseñaron *prompts* específicos que instruyen al modelo para que realice una transcripción fiel del contenido de la imagen, preservando símbolos, mayúsculas, minúsculas y errores propios del deterioro. Se evita corregir, completar o inferir palabras ausentes, con el objetivo de evitar alucinaciones por parte del modelo.

Para todos los motores de OCR, el sistema permite aplicar un preprocesamiento opcional. Este preprocesamiento puede ejecutar dos operaciones: recorte de bordes para eliminar márgenes o zonas que no aportan a la transcripción; y segmentación en bloques (párrafos/columnas/secciones), tras la cual cada bloque se transcribe de manera independiente. Cuando la segmentación se selecciona, el *pipeline* recompone la salida respetando el orden de lectura del documento. Luego, las transcripciones generadas se almacenan en archivos de salida.

El Capítulo 5 profundiza en el trabajo realizado en este módulo, incluyendo enfoques, experimentos, resultados y análisis.

No se realizan correcciones ni preprocesamiento en la salida OCR. La limpieza

consiste únicamente en la extracción del texto transcrito, sin ningún mensaje adicional, como los que suelen incluirse por el LLM.

4.2.2. Módulo NER

Este módulo recibe como entrada un texto plano y su salida es una lista de etiquetas, donde se asigna a cada palabra una posible clasificación.

Por motivos de alcance, solo se trabajó con modelos del tipo BERT, entrenados específicamente para detectar entidades en texto en español. La elección se fundamenta en que, como se menciona en la Subsección 2.7.2, la familia de modelos BERT es ampliamente utilizada para resolver esta tarea, con resultados que determinan el estado del arte. Adicionalmente, dentro de la familia *Transformer*, estos son relativamente livianos, evitando un aumento excesivo del tiempo de ejecución.

El entrenamiento utiliza tanto datos reales como un conjunto de ejemplos sintéticos generados por un módulo propio, en el que se generan ejemplos artificiales en el dominio mediante el uso de LLMs, con anotaciones BIO. Su objetivo es aumentar la cobertura de combinaciones entidad-contexto y mejorar el rendimiento de NER.

Dado que las transcripciones de OCR introducen artefactos, se incorpora un módulo que agrega ruido a los ejemplos sintéticos a través de patrones y funciones, con el fin de simular las imperfecciones presentes en los datos reales. Entre ellas se incluyen sustituciones de caracteres y la segmentación de palabras.

Estas últimas son tratadas como ciudadanas de primera clase[2], aplicándose sobre las cadenas siguiendo un paradigma similar al funcional; esto permite complejizar los patrones de ruido e introducir aleatoriedad.

Todo lo referente al módulo NER se detalla en el Capítulo 6.

4.3. Arquitectura del sistema

El sistema propuesto en el presente proyecto modifica lo planteado en los antecedentes locales (ver Sección 3.1). En particular, se reemplaza la etapa de *Preprocesamiento del texto* por una de OCR, ya que se trabaja con imágenes en lugar de transcripciones previas de LUISA.

A continuación, se presenta brevemente un patrón de diseño que ha resultado efectivo para resolver problemas similares en el pasado. El uso de patrones y buenas prácticas favorece la reutilización y la mantenibilidad del código.

[2] https://en.wikipedia.org/wiki/First-class_citizen

Patrón estrategia

El desarrollo de módulos intercambiables se adapta al patrón de diseño de comportamiento *«Estrategia»* (Gamma, Helm, Johnson, y Vlissides, 1994). Este permite seleccionar un algoritmo en tiempo de ejecución en lugar de implementar un único algoritmo directamente; el código recibe instrucciones en tiempo de ejecución sobre cuál de un conjunto de algoritmos utilizar.

Esto es útil, por ejemplo, para modelar cada posible implementación del OCR como una estrategia (*Tesseract*, *Llama-V*, etc.), guiando el desarrollo y brindando un marco de referencia. La Figura 4.3 muestra una situación genérica de aplicación del patrón.

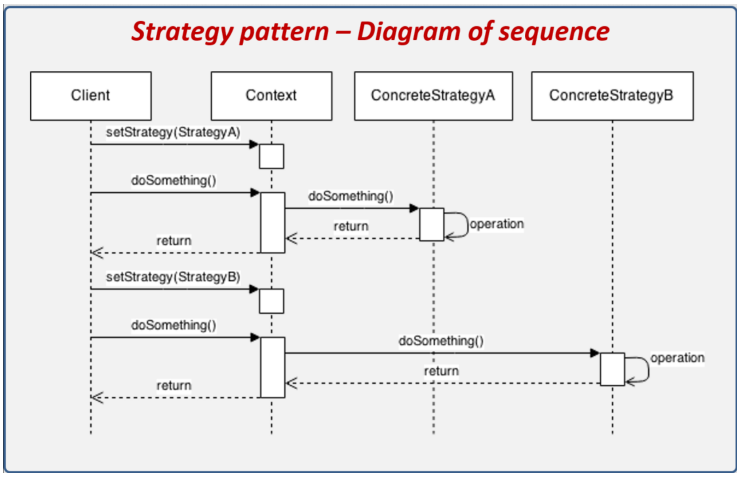


Figura 4.3: Diagrama de secuencia que representa el patrón de diseño Estrategia[3].

[3]Strategy: https://reactiveprogramming.io/blog/en/design-patterns/strategy

Diagrama de arquitectura

A modo de resumen, se presenta la Figura 4.4, que muestra de forma esquemática la arquitectura completa del prototipo.

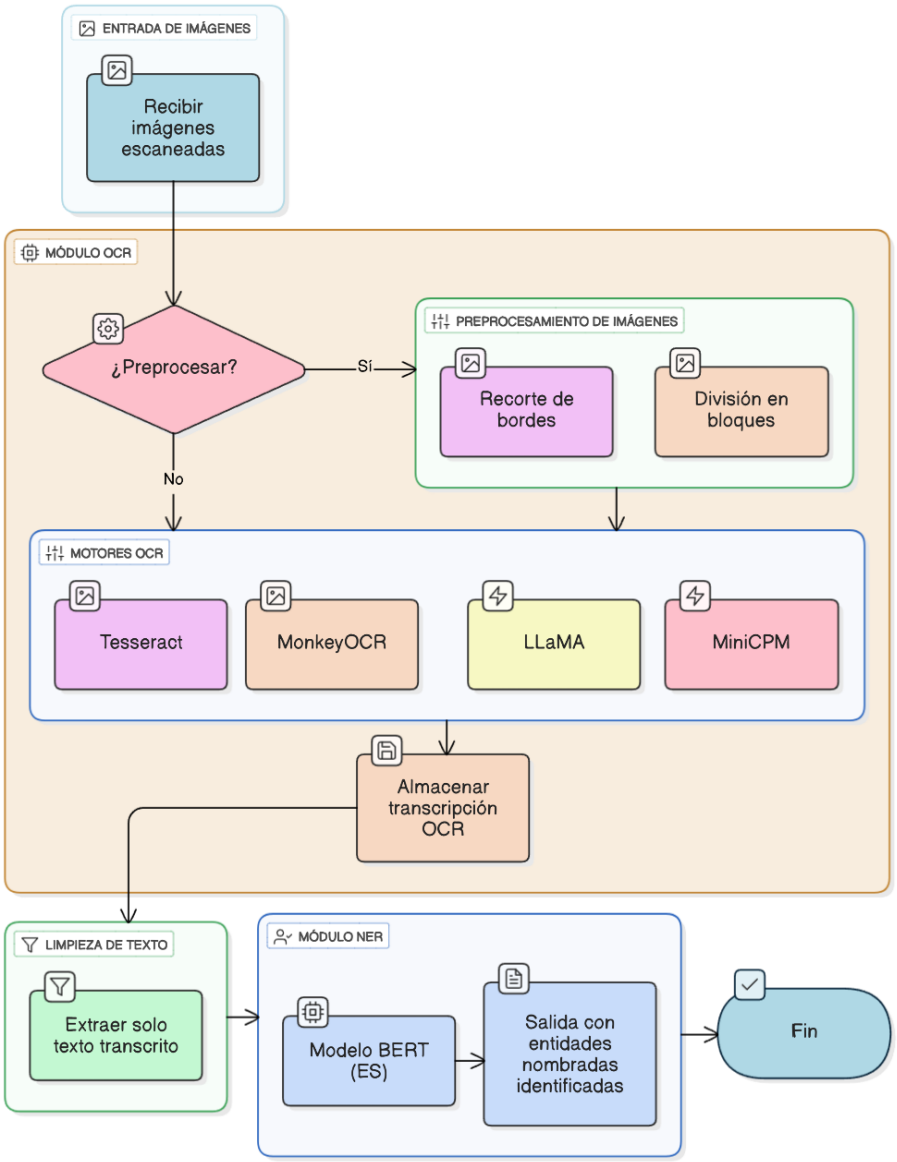


Figura 4.4: Esquema de arquitectura del prototipo.

El flujo incluye un punto de decisión para aplicar o no un preprocesamiento

(recorte de bordes y/o segmentación en bloques) antes de elegir el motor OCR que realizará la transcripción. La salida se almacena y luego se posprocesa, donde se le aplica una limpieza del texto. Finalmente, el texto plano es procesado por el modelo NER, en el que las entidades son identificadas en formato BIO.

4.3.1. Aplicación web para la validación del *pipeline*

En línea con el Objetivo General, se desarrolló una aplicación que permite ejecutar y validar el *pipeline* de punta a punta sobre documentos reales. El propósito es tener un entorno de experimentación, donde sea sencillo comparar motores de OCR y modelos NER y observar los resultados. La Figura 4.5 muestra el diseño de la página.

La aplicación acepta una o varias imágenes a ser procesadas, donde se incluye una interfaz que permite seleccionar el motor de OCR. Luego de ser transcritas, pueden descargarse en distintos archivos de texto para cada imagen o continuar la ejecución de la etapa NER. Además, en la sección de estadísticas se puede visualizar la proporción de documentos procesados correctamente.

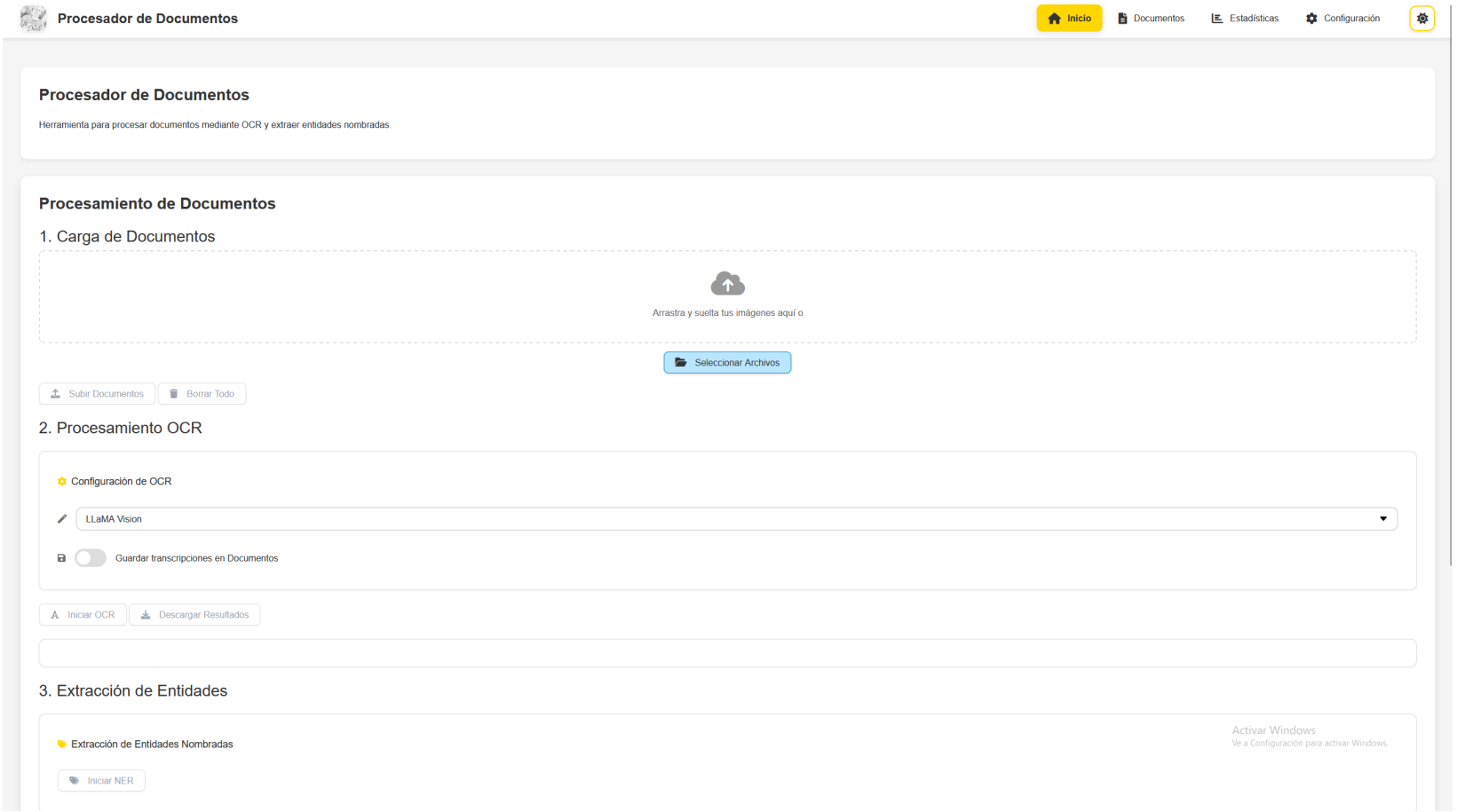


Figura 4.5: Vista principal de la aplicación

Capítulo 5

Módulo OCR

El reconocimiento óptico de caracteres es uno de los primeros pasos en el proceso de extracción de información y sus resultados constituyen el principal insumo de todas las tareas posteriores.

Múltiples investigaciones recientes indican que la calidad del proceso de OCR tiene un impacto considerable en la resolución de problemas de extracción de información. En particular, el etiquetado de entidades nombradas muestra resultados progresivamente peores a medida que la calidad de las transcripciones disminuye (Van Strien y cols., 2020). La obtención de transcripciones de alta calidad se ve fuertemente afectada por la calidad de las imágenes, que, en el contexto del trabajo, presentan artefactos como sellos, manchas o texto de baja visibilidad.

En los últimos años se han desarrollado múltiples trabajos orientados a mejorar los métodos de OCR aplicados al procesamiento de este conjunto de imágenes. Sin embargo, las transcripciones disponibles actualmente en la plataforma LUZ continúan presentando errores significativos, incluso en documentos de buena legibilidad. En los casos más críticos, el reconocimiento automático falla por completo, produciendo resultados en los que el texto resultante carece de correspondencia con el contenido del documento original, lo que implica una pérdida total de la información presente en la imagen (Stabile, Fernández, y Fioritto, 2020b; Nogueira y cols., 2024b). Por esta razón, se exploraron soluciones alternativas que permitieran mejorar los resultados de transcripción y, en consecuencia, ofrecer una base más robusta para las siguientes etapas del *pipeline*.

Durante la etapa de OCR, se evaluó el desempeño de diferentes tecnologías con el objetivo de obtener transcripciones textuales a partir de documentos históricos deteriorados. Si bien esta etapa no constituye el eje central del proyecto, es de gran importancia, ya que tiene un impacto en los módulos posteriores. Por esta razón, se trabajó en evaluar métodos que permitan obtener transcripciones de mayor calidad que las generadas con *Tesseract* OCR, en particular, utilizan-

do grandes modelos de lenguaje multimodales, con el fin de mejorar el corpus disponible para el sistema de extracción de entidades (Bhadauria, Múnera, y Krestel, 2024).

Las transcripciones existentes, realizadas con *Tesseract*, se tomaron como línea base. La herramienta *Tesseract* es configurable y rápida, pero presenta limitaciones. Debido a su funcionamiento, donde la interpretación del texto es a nivel de carácter, se observan errores como la inserción de caracteres erróneos, la generación de palabras sin sentido, la lectura de ruido visual como texto válido y la segmentación de palabras.

A continuación, se detallan los experimentos realizados tanto con modelos LLM multimodales como con otras herramientas específicas de OCR.

5.1. Uso de modelos multimodales para OCR

Se evalúa el uso de modelos de lenguaje multimodal como LLAMA *Vision* y *MiniCPM*, que presentaron problemas de distinta naturaleza. Inicialmente, los modelos tendían a omitir la transcripción de información sensible, como números de cédula de identidad, probablemente debido a filtros éticos internos incorporados durante su entrenamiento. Para resolver esta limitación, se incorporaron ajustes al *prompt*.

Otro problema frecuente fue la duplicación de palabras cuando aparecían fragmentadas en dos renglones; los modelos tienden a completar el fragmento y luego repetir la palabra completa. Para abordar esta situación, se diseñó un *prompt* (consultar Sección B.2) que instruye estrictamente que el texto debe estar completo; las palabras deben ser transcritas tal como aparecen, sin inferir ni completar términos incompletos. Estas intervenciones buscan preservar la integridad del documento y asegurar que el modelo actúe como un transcriptor fiel, sin introducir modificaciones.

En términos de la configuración de los LLMS, se optó por una estrategia de generación controlada y poco exploratoria. Se implementó *beam search* con cinco *beams* y también se evaluaron configuraciones con temperaturas bajas. Ambas estrategias buscaron equilibrar la exploración de múltiples posibilidades de transcripción sin caer en resultados inconsistentes, asegurando una salida más coherente y menos propensa a errores. Se evitó el uso de *sampling* puro o de temperaturas altas para minimizar la creatividad no deseada en un contexto donde la fidelidad al documento original es prioritaria; como ya se mencionó, es importante que el modelo no introduzca alucinaciones.

5.1.1. Exploración inicial y definición de corpus de prueba

Para evaluar el rendimiento de los modelos, se realizó una serie de pruebas iniciales sobre nueve textos cortos, cada uno con hasta cuatro párrafos. El objetivo

principal de esta etapa exploratoria fue definir las instrucciones proporcionadas al modelo para realizar las transcripciones en un contexto acotado.

Las pruebas se realizaron con imágenes de alta calidad, lo que permitió analizar la capacidad de los modelos para recuperar texto legible en condiciones ideales, sin la influencia de artefactos. Esta instancia inicial también sirvió para identificar errores frecuentes en el comportamiento de los modelos LLM al enfrentarse a tareas de transcripción.

Por ejemplo, durante la transcripción de fragmentos en negrita, los modelos pueden representar este formato tipográfico mediante etiquetas como «br», lo que introduce caracteres no deseados en el resultado final. Otro comportamiento frecuente fue que, ante textos extensos, los modelos suelen centrar la atención en las zonas más nítidas, omitiendo partes del contenido deterioradas. Esta tendencia impacta especialmente a los documentos con los que se trabaja, ya que las imágenes de baja legibilidad son comunes y la pérdida de segmentos completos afecta de manera significativa las métricas de calidad de la transcripción. Este fenómeno imposibilita la extracción de la información objetivada en estos segmentos en etapas posteriores, por lo que evitarlo resulta especialmente relevante.

Es posible, como parte del preprocesamiento, dividir cada imagen a procesar en segmentos, lo que reduce tanto la complejidad de la entrada como la cantidad de texto que el modelo debe transcribir en cada inferencia. Como consecuencia, se mitiga el riesgo de que el modelo omita parte del contenido.

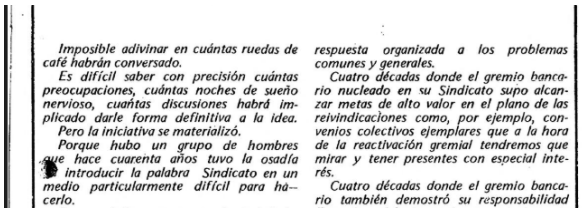
Sin embargo, se identifica que recortar excesivamente la longitud del texto de entrada reduce el contexto disponible para el modelo, lo que dificulta la desambiguación de palabras ilegibles o de fragmentos deteriorados. Por esta razón, en etapas posteriores, para favorecer tanto la calidad de la transcripción como el aprovechamiento del contexto global, se decidió evitar divisiones por renglón o por palabra; en su lugar, se realizaron cortes sobre fragmentos del texto naturalmente delimitados.

Para la generación de un *ground truth* de referencia que se utilizara en la evaluación comparativa, se seleccionó una muestra representativa de 21 imágenes de LUZ, donde se representa la diversidad y dificultad reales encontradas en el conjunto. Estas imágenes corresponden a documentos extensos y de calidad variable, seleccionadas de modo que la distribución fuera equilibrada entre baja, media y alta calidad, según la métrica disponible en la plataforma (Ecuación 5.1).

La calidad de los documentos se define entonces de la siguiente manera:

$$Calidad(p : Puntaje) := \begin{cases} Alta \text{ si } p > 66 \\ Media \text{ si } 33 < p \leq 66 \\ Baja \text{ si } p \leq 33 \end{cases} \tag{5.1}$$

Las Figuras 5.1, 5.2 y 5.3 muestran ejemplos de cada una de las categorías.



Imposible adivinar en cuántas ruedas de café habrán conversado.

Es difícil saber con precisión cuántas preocupaciones, cuántas noches de sueño nervioso, cuantas discusiones habrá implicado darle forma definitiva a la idea.

Pero la iniciativa se materializó.

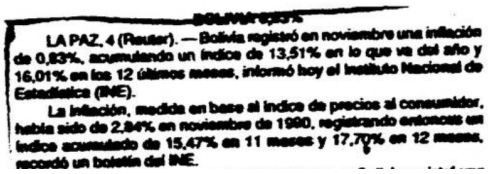
Porque hubo un grupo de hombres que hace cuarenta años tuvo la osadía de introducir la palabra Sindicato en un medio particularmente difícil para hacerlo.

respuesta organizada a los problemas comunes y generales.

Cuatro décadas donde el gremio bancario nucleado en su Sindicato supo alcanzar metas de alto valor en el plano de las reivindicaciones como, por ejemplo, convenios colectivos ejemplares que a la hora de la reactivación gremial tendremos que mirar y tener presentes con especial interés.

Cuatro décadas donde el gremio bancario también demostró su responsabilidad

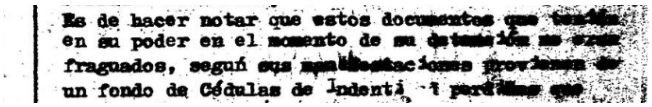
Figura 5.1: Fragmento de documento de calidad alta



LA PAZ, 4 (Reuter). — Bolivia registró en noviembre una inflación de 0,83%, acumulando un índice de 13,51% en lo que va del año y 16,01% en los 12 últimos meses, informó hoy el Instituto Nacional de Estadística (INE).

La inflación, medida en base al índice de precios al consumidor, había sido de 2,84% en noviembre de 1990, registrando entonces un índice acumulado de 15,47% en 11 meses y 17,70% en 12 meses, recordó un boletín del INE.

Figura 5.2: Fragmento de documento de calidad media



Es de hacer notar que estos documentos que tenía en su poder en el momento de su detención no eran fraguados, según sus manifestaciones provenían de un fondo de Cédulas de Identi

Figura 5.3: Fragmento de documento de calidad baja

Este conjunto incluye documentos multicolumna, manuscritos, mecanografiados, elementos gráficos y rotaciones. En ningún caso se incluyen documentos cuya calidad de imagen los haga ilegibles para personas, ya que es necesario realizar una transcripción correcta para utilizarlos como texto de referencia a la hora de evaluar.

Las transcripciones de referencia para estas 21 imágenes se realizaron manualmente, asegurando precisión y fidelidad al contenido original, y estableciendo así la base de comparación para el desempeño de los distintos modelos analizados.

5.1.2. Preprocesamiento de imágenes a transcribir

A partir de las heurísticas identificadas en la etapa exploratoria, se diseñó un preprocesamiento específico para las imágenes de entrada, con el objetivo de facilitar la tarea de transcripción de los modelos de lenguaje. Las imágenes originales presentan ruido y, con frecuencia, márgenes y bordes no relevantes, lo que puede afectar la calidad de la transcripción.

Realizar una segmentación adecuada de la imagen de entrada requiere considerar la estructura general o *layout* del documento. Este problema, denominado *document layout analysis*, incluye identificar y clasificar cuáles son las secciones de interés y determinar el orden de lectura de las mismas. Por ejemplo, los tex-

tos periodísticos suelen emplear un *layout* en dos columnas, por lo que dividir este tipo de documento mediante cortes horizontales genera segmentos que no respetan la estructura lógica del contenido.

Para la resolución de esta tarea se define un módulo que emplea funciones de segmentación implementadas por las librerías *EasyOCR*[1] y *Tesseract OCR*.

En primer lugar, *EasyOCR* provee funcionalidades que permiten eliminar automáticamente los bordes y centrarse únicamente en la región que contiene texto relevante. El procedimiento consiste en detectar las regiones con texto mediante un modelo CRAFT (*Character-Region Awareness For Text*) (Baek, Lee, Han, Yun, y Lee, 2019) y recortar la imagen original de modo que solo conserve el área estrictamente necesaria; de esta manera, se minimiza la presencia de artefactos.

A continuación, se segmenta el texto en secciones para su procesamiento. Se utilizó *Tesseract* OCR, configurado para identificar y extraer los párrafos de cada imagen preprocesada. La implementación permite detectar regiones rectangulares de la imagen de entrada que delimitan un segmento de texto de interés (denominadas *bounding boxes*) en los párrafos, descartando regiones vacías.

Durante este proceso se detectaron errores en la división de las imágenes, en los que se asignaron regiones de texto solapadas a distintos párrafos, generando *bounding boxes* que se intersectan parcialmente. Este incidente ha sido reportado como un problema abierto[2], en el que una misma línea o palabra puede incluirse en más de un párrafo cuando el modelo no logra delimitar correctamente las estructuras.

Para solucionarlo, fue necesario implementar una verificación adicional luego de ejecutar la segmentación de *Tesseract*, agrupando los párrafos cuyos *bounding boxes* se solapan, evitando la duplicación de líneas al generar los recortes individuales.

Una vez realizada la segmentación, los párrafos fueron ordenados respetando la disposición original del documento, agrupando los elementos en función de su cercanía vertical y horizontal en orden de lectura. Finalmente, los párrafos fusionados se almacenaron como imágenes independientes para su posterior procesamiento.

Un inconveniente de este preprocesamiento es que la calidad de la segmentación de la imagen depende en gran medida de la capacidad de *Tesseract* para detectar correctamente el *layout*. Además del problema previamente mencionado, esta herramienta suele presentar dificultades para identificar estructuras multicolumna en textos ruidosos, lo que puede derivar en una transcripción fiel de cada bloque individual, pero con un orden de lectura incorrecto.

[1] EASYOCR:https://github.com/JaidedAI/EasyOCR

[2] BUG: TESSERACT PRODUCES OVERLAPPING BOUNDING BOXES FOR CLEARLY SEPARATED LINES: https://github.com/tesseract-ocr/tesseract/issues/3963?utm_source=chatgpt.com

Es importante mencionar que, al emplearse un procesamiento de división fuertemente basado en la implementación de *Tesseract*, los defectos de división tienden a presentarse de forma similar en las herramientas que comparten esta dependencia.

5.1.3. Evaluación comparativa de métodos de OCR

A partir del conjunto de documentos seleccionados, se evaluó el rendimiento de los modelos de lenguaje con dos configuraciones, imágenes completas con y sin preprocesamiento, donde se realiza la división de las imágenes en bloques. Para la evaluación se consideran las métricas WER 2.3, CER 2.2, BLEU 2.4 y CHARF 2.7, presentadas en el capítulo 2.

En la Figura 5.4 se presentan los resultados de la evaluación de los distintos modelos sobre cada subconjunto de los datos de evaluación, incluyendo *Tesseract* y los modelos multimodales *Llama 3.2-11B-Vision*[3] y *MiniCPM-o 2.6*[4]. Cada modelo multimodal es evaluado con y sin aplicar preprocesamiento. Inicialmente, el análisis se enfoca exclusivamente en el desempeño de las variantes que utilizan LLMs, antes de considerar comparaciones con *Tesseract*.

Para este análisis, los documentos de evaluación se agrupan en tres niveles de dificultad en función de su calidad: los de baja calidad se consideran parte de la categoría «fácil», los de calidad media se asignan a la categoría «medio» y los de alta calidad a la categoría «difícil». Esta segmentación permite observar cómo varía el desempeño de los modelos según la degradación del documento.

Dado que las salidas serán posteriormente utilizadas por el módulo NER, es importante mantener una fidelidad carácter a carácter, ya que errores mínimos pueden impactar negativamente en la correcta detección de entidades.

Se observa que, para documentos de la categoría «fácil», el modelo *MiniCPM* alcanza las medianas más bajas tanto para CER como para WER, lo que sugiere que la segmentación por bloques es efectiva en este escenario. Por otro lado, a medida que aumenta la dificultad del texto, esta ventaja tiende a atenuarse; en documentos de dificultad media o alta, el procesamiento de la página completa llega a resultados similares o incluso superiores a los de la versión segmentada. Esto indica que, en contextos más complejos, la segmentación puede interferir con la estructura de la entrada, lo que afecta negativamente el rendimiento del modelo.

Con respecto a *Tesseract*, es la herramienta con el mejor rendimiento en documentos de baja dificultad. En esta categoría, supera a los LLMS evaluados tanto en la métrica CER como en WER, lo que indica una alta precisión a nivel de carácter y de palabra. Además, al considerar las métricas globales (CHRF y BLEU), los resultados de *Tesseract* se mantienen similares a los obtenidos por los

[3]Llama 3.2-11B-Vision: https://huggingface.co/meta-llama/Llama-3.2-11B-Vision
[4]MiniCPM-V: https://github.com/OpenBMB/MiniCPM-o

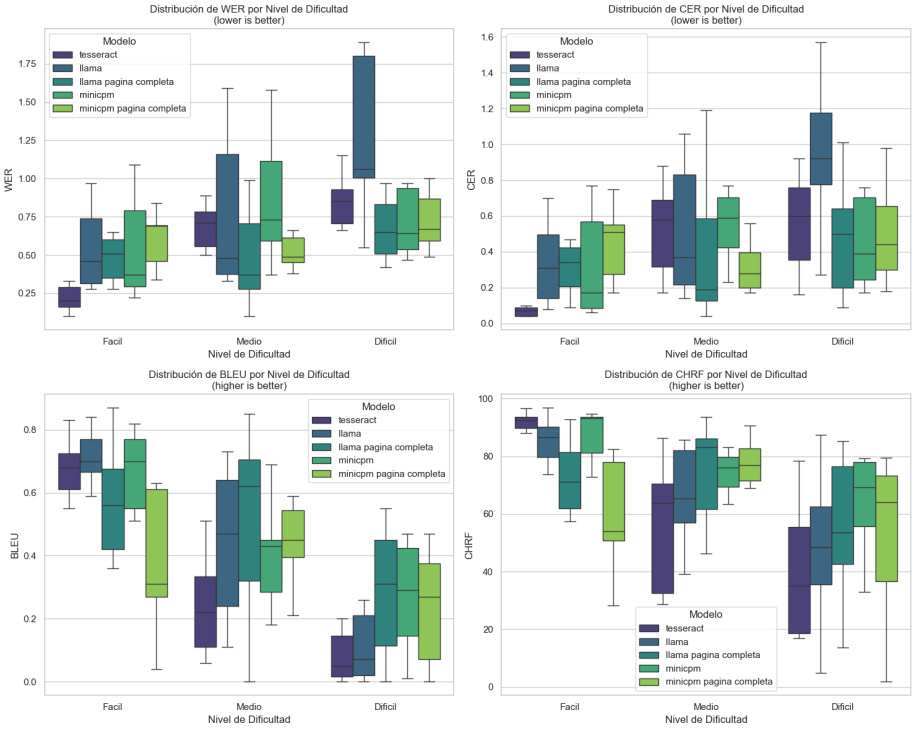


Figura 5.4: Métricas *MiniCPM* y LLAMA con división por bloques vs página completa

modelos. Esto indica que, al menos para textos simples, *Tesseract* sigue siendo la mejor solución.

Por otro lado, el comportamiento de los LLMS es inverso: a medida que la dificultad aumenta, el rendimiento es superior al de *Tesseract*. En documentos de dificultad media, LLAMA en página completa logra los mejores resultados en todas las métricas. Para documentos de dificultad alta, a excepción de LLAMA con segmentación de documento, las métricas de los LLMs superan ampliamente a las de *Tesseract*.

En conclusión, la segmentación resulta beneficiosa únicamente con contextos de baja complejidad, donde la simplicidad del *layout* reduce el riesgo de introducir errores en los bloques. Por otro lado, cuando el texto presenta una estructura compleja o ruido, conservar la página entera permite a los modelos aprovechar el contexto completo y mantener un buen rendimiento. Por otro lado, los tiempos de inferencia de *Tesseract* son mucho menores que los de cualquiera de las variantes que utilizan LLMS, por lo que es la herramienta preferida para documentos de complejidad baja.

En cuanto al tiempo de inferencia, se observa que las versiones que procesan la página completa son las que presentan el menor tiempo de ejecución. Esto hace que sean las opciones más convenientes no solo desde el punto de vista del rendimiento general del modelo, sino también en términos de eficiencia. Esto tiene mayor relevancia en el contexto del proyecto, ya que se trabaja con una gran cantidad de documentos. Por lo tanto, reducir el tiempo de inferencia por documento tiene un alto impacto en los tiempos totales de procesamiento, lo que permite que la solución sea escalable.

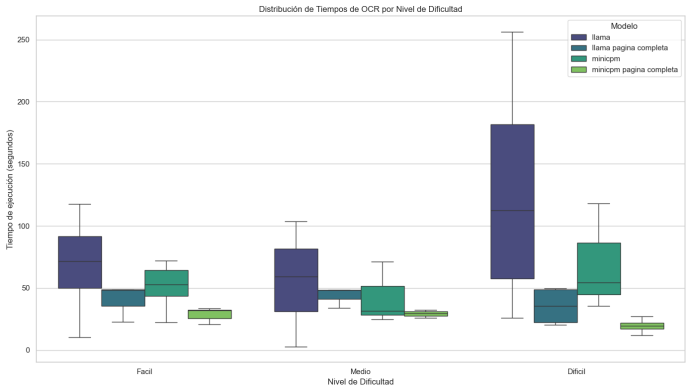


Figura 5.5: Tiempo de inferencia por dificultad

5.1.4. Evaluación con herramientas específicas para OCR

Con el objetivo de comparar diferentes aproximaciones de OCR, se evaluó también el rendimiento de *MonkeyOCR*, un sistema que introduce el paradigma «Structure-Recognition-Relation» (SRR) para el análisis de documentos. Este enfoque simplifica el *pipeline* de procesamiento al resolver distintas tareas, evitando el uso de LLMs para procesar páginas completas. El problema se aborda buscando la respuesta a tres preguntas: ¿dónde se encuentra el contenido? (*Structure*), ¿qué contiene cada bloque? (*Recognition*) ¿Y cómo está organizado? (*Relation*) relacionado con el orden lógico de lectura. Con esta implementación se simplifica el *pipeline* y es más eficiente computacionalmente, ya que evalúa la página por secciones. Luego de la segmentación del documento, se pueden emplear distintos LLMs para transcribir la página (Liu y cols., 2024).

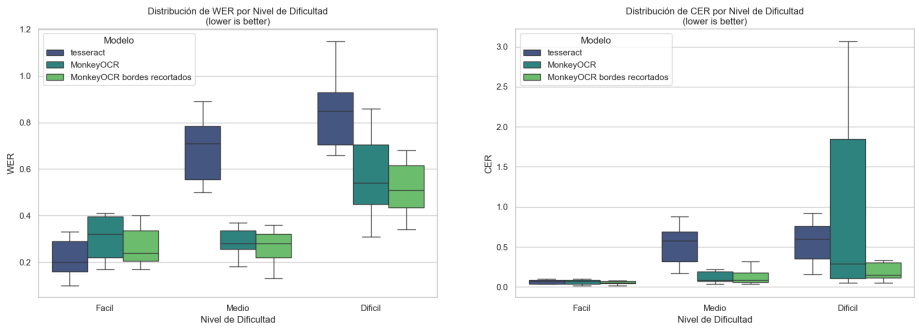
Para la transcripción, se utilizó como modelo de reconocimiento de texto a *Qwen2.5-VL*.

Se aplicó *MonkeyOCR* al mismo conjunto de 21 imágenes, junto con las transcripciones manuales utilizadas para evaluar otros métodos. Se evaluaron tres configuraciones de OCR para la comparación: *Tesseract* y *MonkeyOCR* con la página completa, y *MonkeyOCR* con el recorte de bordes utilizado por *EasyOCR*. Este último se evaluó con el fin de evaluar si es posible reducir los

tiempos de inferencia utilizando páginas de menor tamaño. La evaluación se realizó mediante las métricas WER, CER, BLEU y CHRF. Al igual que en la sección anterior, los resultados se agruparon según el nivel de dificultad estimado de cada documento (fácil, medio o difícil).

Dado que *MonkeyOCR* utiliza un LLM para la transcripción, presenta problemas similares a los mencionados en la sección anterior, como la repetición de texto o la dificultad para transcribir todo el texto de la imagen. Sin embargo, estos problemas se atenuan gracias al procesamiento previo aplicado a las imágenes antes de la inferencia. *Tesseract* realiza la transcripción a nivel de carácter, lo que puede facilitar valores más altos de CER. Por otro lado, *MonkeyOCR* presenta una ventaja en la métrica WER, ya que generalmente los LLM tienen la capacidad de desambiguar palabras a partir del contexto, permitiendo deducir palabras incluso si uno o varios caracteres son ilegibles.

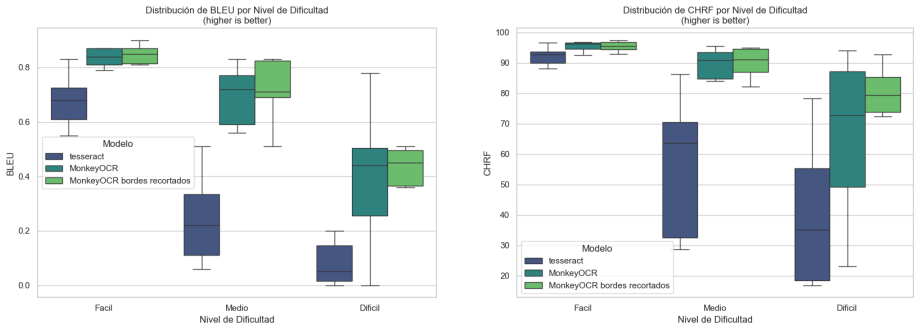
Como ilustran las figuras 5.6a y 5.6b, *Tesseract* presenta un rendimiento superior en los documentos de menor dificultad. Sin embargo, en imágenes de dificultad media o alta, el modelo basado en *Transformer* presenta un rendimiento notablemente superior. Se observa que la WER en el intervalo intercuartil es estrictamente menor en las transcripciones realizadas por *MonkeyOCR*. Se nota también que el recorte de márgenes resulta en una mejora de menor magnitud, pero consistente, en ambas métricas y en los tres conjuntos evaluados. Este preprocesamiento presenta también una mejora en la consistencia de los resultados, especialmente en documentos de alta dificultad. Esto es especialmente notable en la métrica CER del conjunto «difícil», donde se aprecia que el rendimiento de los cuartiles altos mejora sustancialmente.



(a) WER (b) CER

Figura 5.6: *MonkeyOCR* vs *Tesseract*: comparación de WER y CER.

El rendimiento de *MonkeyOCR* también es consistentemente superior según las mediciones de BLEU y CHRF, incluso en documentos de nivel «fácil», donde *Tesseract* generalmente alcanza buenos resultados. Para los niveles «fácil» y «medio», las diferencias entre el uso o no del recorte de bordes en *MonkeyOCR* son menores. Sin embargo, de forma consistente con los análisis de WER y CER en documentos difíciles, el recorte de borde contribuye a una mejor calidad y a una mayor consistencia en las inferencias. Esto es especialmente notable a nivel de carácter, donde casi la totalidad de las transcripciones muestra un CRHF mayor que la mediana del conjunto de transcripciones donde los bordes no fueron eliminados.



(a) CHRF (b) BLEU

Figura 5.7: *MonkeyOCR* vs *Tesseract*: comparación de CHRF y BLEU.

Al realizar un análisis por documento, se observa que en algunos casos particulares el recorte de bordes es altamente relevante para el correcto procesamiento de la imagen.

El siguiente análisis se basa únicamente en la métrica BLEU, ya que sus resultados son representativos del comportamiento general observado en el resto de las métricas.

La Figura 5.8 ilustra el rendimiento con granularidad a nivel de documento con el objetivo de identificar si la mejora al aplicar el recorte es general o si existen particularidades que afecten negativamente a los resultados.

Los documentos 11 y 18 son los que presentan las mayores diferencias. En ambos casos, se trata de fichas completadas con texto manuscrito. El documento 10 comparte estas características, pero no muestra una mejora significativa, lo que sugiere que el impacto del recorte no es uniforme en textos manuscritos. Además, los bordes de estas fichas son amplios y contienen sellos, por lo que su eliminación podría contribuir a mejorar el rendimiento del OCR.

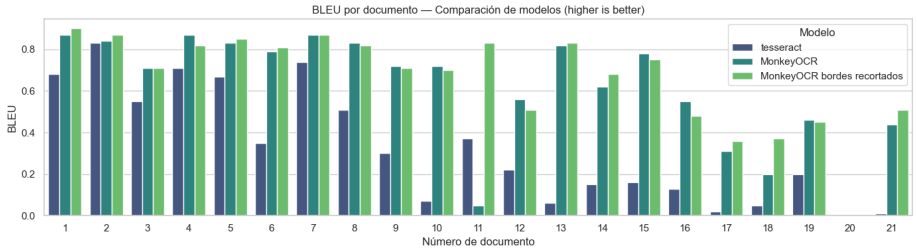


Figura 5.8: Métrica BLEU para *MonkeyOCR* y *Tesseract* por documento

En cuanto al tiempo de procesamiento, se observa una diferencia significativa entre las ejecuciones con y sin recorte de bordes. Es importante tener en cuenta que el tiempo reportado para el OCR con recorte incluye tanto la etapa de recorte de bordes como la transcripción.

El tiempo total requerido para procesar todos los documentos con recorte de bordes fue de 1103,8 segundos, mientras que el mismo procesamiento sin recorte de bordes demora 1374,6 segundos. Esto implica una reducción de casi el 20 % en el tiempo total cuando se aplica el recorte, lo que representa una mejora relevante en términos de eficiencia.

La Figura 5.9 muestra el tiempo de procesamiento por documento. La mejora observada es particularmente notoria en documentos que, sin el recorte, presentan regiones de borde con ruido o áreas no relevantes, un escenario común en los datos de LUZ, los cuales pueden incrementar innecesariamente el tiempo de inferencia del modelo SRR.

En ciertos casos, la diferencia puede ser menos pronunciada. Estos resultados están asociados a documentos que ya poseen una estructura bien delimitada, sin bordes excesivos o artefactos, por lo que el beneficio del recorte es marginal.

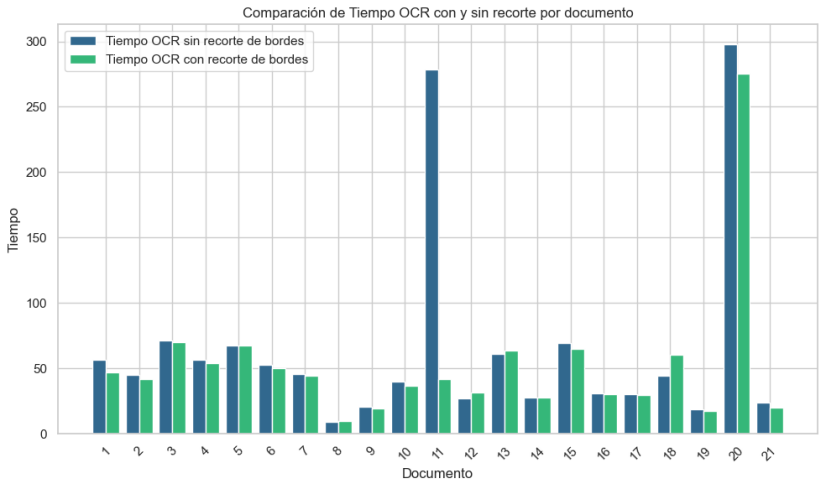


Figura 5.9: Tiempo de procesamiento por documento

En conclusión, la incorporación del recorte de bordes como etapa de preprocesamiento no solo mejora las métricas de calidad de inferencia en ejemplos complejos, sino que también optimiza los recursos computacionales, reduciendo el tiempo total de procesamiento de grandes volúmenes de documentos. Esto es muy relevante en el contexto de aplicación de la herramienta, donde el volumen de datos a procesar es elevado y la eficiencia en cuanto a tiempo es crítica.

5.2. Conclusiones

El análisis desarrollado en este capítulo permitió identificar las limitaciones de los métodos de OCR tradicionales, en particular *Tesseract OCR*, y contrastarlas con enfoques recientes basados en modelos multimodales y en arquitecturas específicas, como *MonkeyOCR*. Los resultados muestran que, aunque TESSERACT continúa siendo competitivo en documentos de baja dificultad y con tiempos de inferencia bajos, los modelos basados en LLMs y *MonkeyOCR* alcanzan un desempeño superior en escenarios de complejidad media y alta, tanto a nivel de palabras como de caracteres. Además, se observó que preprocesamientos, como el recorte de bordes, no solo mejoran la fidelidad a las transcripciones, sino que también reducen los tiempos de procesamiento.

En síntesis, el trabajo realizado en esta etapa permitió obtener transcripciones más precisas y consistentes, mejoras fundamentales para las etapas posteriores del *pipeline*. En particular, las mejoras en la calidad de las transcripciones impactan directamente en la capacidad del sistema para realizar NER. De esta forma, esta etapa establece las bases necesarias para abordar las tareas del siguiente capítulo.

Capítulo 6

Módulo NER

Uno de los objetivos del proyecto es construir un sistema de reconocimiento de entidades con nombre. Para cumplir con este objetivo, es necesario contar con conocimientos actualizados en este tema; las secciones 2.3 y 2.7 brindan un panorama detallado de los conceptos teóricos y técnicos del estado del arte.

Este capítulo describe el desarrollo de las herramientas implementadas, la preparación de los conjuntos de datos utilizados, así como los resultados obtenidos.

6.1. Obtención y descripción del corpus

El corpus anotado para entrenar los modelos de NER se construyó a partir de dos conjuntos de datos generados en el marco del proyecto CRUZAR. En ambos casos, los datos se segmentaron en secuencias cortas, procurando que las entidades no quedaran divididas entre fragmentos.

La primera fuente corresponde a un conjunto elaborado durante el proyecto de grado mencionado en la Sección 3.1 (Cardozo Ramírez y cols., 2021). Este conjunto de datos utiliza el esquema de etiquetado BIO e incluye las etiquetas PER (personas), ORG (organizaciones), LOC (lugares) y DATE (fechas). Luego del proceso de anotación, el conjunto fue curado por los participantes del proyecto LUZ.

La segunda fuente proviene de un módulo de extensión en el que los estudiantes realizaron anotaciones sobre un conjunto reducido de documentos. En este caso, el conjunto de datos incluye, además de las categorías anteriores, la de *sujeto omitido* y incorpora anotaciones de correferencias. El corpus está compuesto por nueve documentos, anotados por dos participantes. En el presente proyecto se llevó a cabo la curación y preparación de los datos antes de integrarlos al conjunto final.

Como se menciona en la Sección 2.7, en tareas clásicas de NER es común utilizar

el esquema BIO, donde cada *token* se clasifica como inicio («B», por «*Begin*»), interior de una entidad («I», por «*Inside*») o fuera de una entidad («O», por «*Outside*»). Para la tarea NER con texto ruidoso optamos por simplificar el esquema de etiquetado. La decisión se fundamenta en la naturaleza del corpus empleado, donde aparecen palabras correspondientes a la misma entidad fragmentadas debido a errores de OCR u oclusiones en los documentos (por ejemplo: «Mar» e «ía» en lugar de «María»).

Para facilitar la tarea de aprendizaje, se utilizó una codificación simplificada donde cada *token* perteneciente a una entidad es etiquetado únicamente con la clase correspondiente (*PER*, *ORG*, *LOC* u *O*), prescindiendo de los prefijos «B-» e «I-». Así, en el ejemplo anterior, tanto «Mar» como «ía» serían etiquetados como «*PER*». Esto permite al modelo centrarse en la identificación de entidades y minimizar el impacto de los errores de segmentación en la evaluación. Esta decisión se basó en las experiencias previas del grupo PLN de la Facultad de Ingeniería al resolver esta tarea con LLMs.

Por otro lado, la simplificación del etiquetado introduce una limitación: al eliminar los prefijos «B-» e «I-», el modelo no puede distinguir entidades contiguas del mismo tipo. Esta ambigüedad afecta principalmente a las métricas a nivel de entidad por fusión de entidades. La aparición de entidades contiguas es poco común, pero puede deberse a errores de OCR (pérdida de conectores o de comas). Utilizamos las etiquetas simplificadas para reducir la complejidad, con la hipótesis de que esta ambigüedad se presentaría con baja frecuencia.

Finalmente, ambos conjuntos fueron estandarizados en formato y en etiquetas. Para este proyecto se conservaron únicamente las categorías PER, ORG y LOC, descartando las adicionales. El corpus, posteriormente, se dividió en tres particiones: 40 % destinado al conjunto de entrenamiento, 30 % para validación y 30 % para prueba. Esta decisión se tomó debido a la cantidad relativamente limitada de datos disponibles, con el objetivo de garantizar que tanto el conjunto de validación como el de prueba cuenten con suficientes ejemplos para obtener resultados representativos y métricas confiables.

6.1.1. Distribución de etiquetas

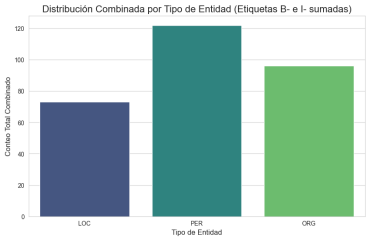
Antes de entrenar el modelo NER, es importante examinar la composición del conjunto de datos para identificar posibles sesgos y patrones que puedan influir en el aprendizaje. El análisis se centró en el conjunto de entrenamiento, evaluando tanto la frecuencia relativa de cada etiqueta como la forma en la que se distribuyen y coexisten.

Los resultados muestran que alrededor del 90 % de los *tokens* están etiquetados como O (sin entidad). A su vez, el 49 % de los ejemplos del conjunto está compuesto únicamente por este tipo de *tokens* (ver Figura 6.1b).

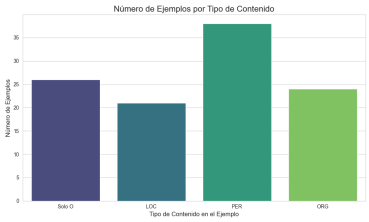
El conjunto de datos de entrenamiento contiene 111 entidades nombradas, distribuidas entre las tres clases estudiadas. La clase más frecuente es PER, que

representa 46 % del total, seguida por ORG con 28 % y, finalmente, LOC con 26 %, lo cual también puede observarse de forma agregada en la Figura 6.1a.

También se observan diferencias en la longitud promedio de las entidades: PER tiene en promedio 2,4 *tokens*, LOC 2,5 y ORG 3,1. Aunque las diferencias son pequeñas, pueden influir en la dificultad del reconocimiento, ya que las entidades más extensas tienden a aumentar la probabilidad de errores de segmentación.



(a) Distribución combinada por tipo de entidad.



(b) Número de ejemplos por tipo de contenido.

La matriz de coocurrencias (Figura 6.2) indica cuántas veces dos tipos distintos de entidades aparecen en un mismo ejemplo. Se observa que las coocurrencias son poco frecuentes, predominando los casos en los que aparece un solo tipo de entidad. Este patrón sugiere que, en el contexto de la aparición de cada clase, tiende a estar separado, lo cual podría simplificar la clasificación, pero limitar la capacidad del modelo para aprender relaciones entre diferentes tipos de entidades.

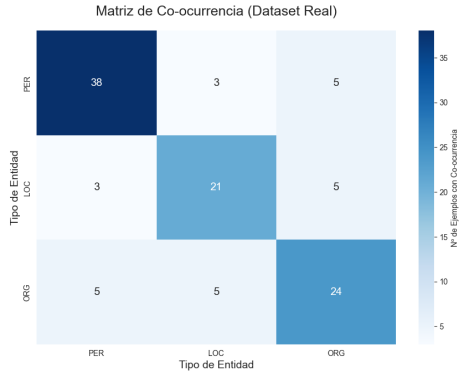


Figura 6.2: Matriz de coocurrencias de datos reales etiquetados.

6.2. Método de evaluación

Antes de comenzar el entrenamiento de los modelos, es necesario definir los criterios de evaluación y las herramientas a utilizar.

El rendimiento de los modelos se evaluó según las métricas de precisión (*precision*), cobertura (*recall*) y $F1_{macro}$, definidas en la Subsección 2.7.3.

La evaluación se realizó bajo un criterio de coincidencia exacta, por lo que se considera verdadero positivo (TP) únicamente si el modelo identifica correctamente todos los *tokens* que componen una entidad. En el caso de que al menos un *token* de la entidad sea clasificado de forma incorrecta, la predicción se considera un falso negativo (FN). Por el contrario, cualquier secuencia etiquetada por el modelo como entidad, pero que no coincida exactamente con la anotación de referencia, se considera un falso positivo (FP).

Este criterio resulta relevante en este contexto, dada la presencia de ruido y fragmentación en el corpus, debidos a errores de OCR y a oclusiones en los documentos. Reconocer entidades incompletas puede provocar que una misma entidad sea almacenada como varias entidades distintas o que se pierda información (por ejemplo, el apellido de una persona), lo que dificulta los siguientes pasos de extracción de información.

6.2.1. Métodos de inferencia

Como se mencionó en la Subsección 2.7.2, si bien se conocen los *tokens* que componen cada palabra, los modelos utilizados infieren una etiqueta (potencialmente distinta) para cada uno de ellos. Dado que se desea asignar una etiqueta a cada palabra, es necesario deducir, a partir de las clasificaciones de sus *tokens*, una única etiqueta por palabra.

El criterio más estricto consiste en asignar una etiqueta solo cuando todos los *tokens* coinciden en la clasificación. Si no todas las etiquetas corresponden al mismo tipo de entidad, se le asigna la etiqueta O. Este es el método utilizado para los análisis sobre validación de los distintos modelos.

Por otro lado, la forma más simple de determinar la clasificación del primer *token*. Este método puede reducir la cantidad de palabras etiquetadas como O, ya que se ignoran las discrepancias internas entre los *tokens*.

6.3. *Baseline*

Se selecciona la herramienta *Stanza* para español como línea base de referencia. Esta herramienta tiene dos versiones, una entrenada con el conjunto de datos *AnCora* y otra con *CoNLL-2002*. El proyecto mencionado en la sección Sección 3.1 reportó un mejor desempeño al utilizar la versión entrenada con el conjunto de datos *AnCora* (Cardozo Ramírez y cols., 2021).

Este modelo fue evaluado en el conjunto de validación previamente definido, aplicando la evaluación a nivel de entidad y las métricas mencionadas. En el conjunto de datos utilizado en el presente proyecto, el modelo entrenado en *CoNLL-2002* muestra un mejor rendimiento que el entrenado en *AnCora*, por lo que se selecciona como *baseline*.

Los autores de *Stanza* reportan una $F1_{macro}$ de 0,88 sobre el corpus *CoNLL-2002* en español (Qi, Zhang, Zhang, Bolton, y Manning, 2020). Los resultados obtenidos al evaluar el conjunto de validación con *Stanza* muestran un rendimiento inferior en comparación con lo esperado para un corpus de calidad alta.

El modelo alcanza un $F1_{macro}$ de 0,39 en validación. El rendimiento por clase muestra que las entidades de tipo PER son las mejor reconocidas, con un $F1_{macro}$ de 0,51, mientras que para ORG se obtiene 0,37 y para LOC 0,29. El modelo logra identificar con mayor confianza los nombres de personas, pero presenta dificultades mayores al identificar organizaciones y lugares. Esta diferencia puede deberse a que muchas organizaciones y lugares son entidades con una gran cantidad de *tokens*, y además, no aparecen o aparecen con poca frecuencia en los datos con los que se entrenó el modelo (*CoNLL-2002*).

Tabla 6.1: Resultados del modelo *Stanza* en el conjunto de validación, usando los modelos entrenados en CoNLL-2002 y en AnCora

Métrica	**Stanza (CoNLL-2002)**	**Stanza (AnCora)**
F1 Macro	**0.28**	0,14
Accuracy	**0.87**	0,82
Precision PER	**0.48**	0,00
Recall PER	**0.44**	0,00
F1 PER	**0.51**	0,00
Precision ORG	**0.36**	0,34
Recall ORG	0,38	**0.53**
F1 ORG	0,37	**0.41**
Precision LOC	0,29	**0.36**
Recall LOC	**0.29**	0,21
F1 LOC	**0.29**	0,26

Aunque *Stanza* sirve como referencia para establecer un punto de comparación inicial, sus resultados en este contexto muestran un bajo rendimiento en comparación con el reportado para un conjunto de datos de alta calidad. Esta brecha justifica la necesidad de explorar modelos más robustos, adaptando modelos preexistentes a un contexto de datos ruidosos.

6.4. Entrenamiento de modelos basados en BERT

Como primer enfoque para resolver la tarea NER, se entrenó un modelo con el corpus previamente definido siguiendo los criterios y métricas descritos en el

estado del arte (Subsección 2.7.3).

Se utilizó *Optuna* (Subsección 2.5.2) para optimizar hiperparámetros, con el objetivo de determinar un conjunto de valores que optimicen la estrategia de entrenamiento. La búsqueda entrena modelos con distintas configuraciones utilizando *early stopping*; se almacena el modelo que muestra el mejor rendimiento ($F1_{macro}$) en validación. En la Tabla 6.2 se presentan los hiperparámetros evaluados y el espacio de búsqueda considerado.

Tabla 6.2: Hiperparámetros evaluados mediante *Optuna* para el ajuste del modelo NER

Hiperparámetro	**Valores considerados**

`Modelo Base`	`FacebookAI/xlm-roberta-large-finetuned-conll02-spanish`
	`dccuchile/bert-base-spanish-wwm-cased`
	`mrm8488/bert-spanish-cased-finetuned-ner`
	`MMG/xlm-roberta-large-ner-spanish`

`Tamaño de lote`	1, 2, 4, 8, 16, 32
`Entrenar sólo última capa`	True, False
`Tasa de Aprendizaje`	$[1 \times 10^{-6}, 5 \times 10^{-4}]$

Los modelos base seleccionados provienen de *Hugging Face*[1] y fueron seleccionados por su uso en tareas de NER en español.

Se ajustaron parámetros comunes en el entrenamiento de modelos de aprendizaje automático como el *learning rate* y el *batch size*.

Adicionalmente, se incorporó la posibilidad de entrenar únicamente la última capa del modelo, controlada con el hiperparámetro *head-only*. Cuando se configura en *True*, solo los parámetros de la última capa se actualizan, manteniendo congeladas las capas anteriores. Esto permite conservar el conocimiento general adquirido durante el preentrenamiento y especializar únicamente el mapeo final de los *embeddings* a las etiquetas de la tarea específica NER para la que se está utilizando.

Por otro lado, al configurarlo como *False*, se habilita el ajuste de todos los parámetros del modelo. Si bien este enfoque permite una mayor capacidad de adaptación, aprendiendo relaciones más complejas sobre el orden y la fragmentación de las etiquetas, también incrementa el costo computacional y el riesgo de sobreajuste, especialmente en escenarios con datos limitados.

La preferencia de *Optuna* de actualizar o no todos los pesos de los modelos base permite obtener información sobre el impacto de los ejemplos en el entrenamiento del modelo.

[1] https://huggingface.co/

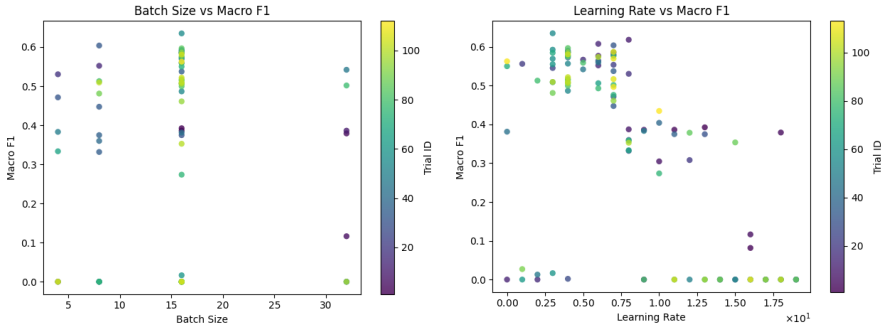
El proceso de optimización permitió identificar configuraciones que maximizan el rendimiento en este contexto.

6.4.1. Análisis de resultados

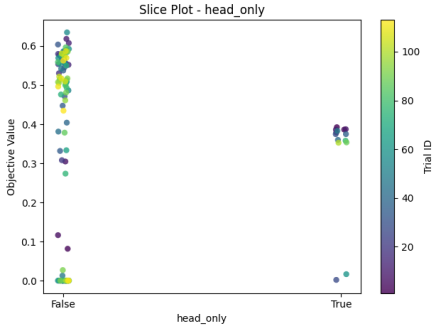
Dado el espacio de experimentación ya definido, se presenta un análisis del rendimiento de los modelos ajustados mediante la optimización de hiperparámetros.

La configuración más efectiva alcanzó un valor de $F1_{macro}$ de 0,63 en el conjunto de validación, con un *batch size* de 16, *head only* en *false* y *learning rate* de 5×10^{-6}.

La Figura 6.9 corresponde a un análisis de *slice plot*, donde se muestra la relación entre el valor objetivo ($F1_{macro}$) y un hiperparámetro específico. Su función es mostrar cómo varía el rendimiento al recorrer el espacio de búsqueda en una sola dimensión, lo que permite identificar tendencias, regiones estables o umbrales de colapso.



(a) *Batch size* vs $F1_{macro}$ (b) *Learning rate* vs $F1_{macro}$



(c) *Head only* vs $F1_{macro}$

Figura 6.3: Resultados de variación de *Batch size*, *Learning rate* y *Head only* sobre $F1_{macro}$.

En general, el modelo logra un mejor rendimiento al utilizar 16 como tamaño de

batch, mientras que *batches* más grandes o más pequeños tienden a degradarlo. Esto tiene sentido considerando la cantidad limitada de ejemplos; con *batches* demasiado grandes, al modelo se le realizan menos actualizaciones por época. Por otro lado, los *batches* muy pequeños aumentan el ruido del gradiente, lo que puede generar soluciones poco estables.

En el *slice plot* del *learning rate* se observa que los valores altos provocan una disminución notoria del rendimiento. Con tasas bajas se logra un mejor equilibrio; el ajuste ocurre de forma progresiva sin alejarse mucho del paso anterior.

El hiperparámetro *head only* muestra una preferencia clara; entrenar solo la última capa reduce el valor de la $F1_{macro}$ de forma consistente. Esto implica que el entrenamiento de la última capa, con el conocimiento previo del modelo, no es suficiente para capturar las particularidades del conjunto de validación. La preferencia de *Optuna* por entrenar todos los pesos del modelo provocó que las ejecuciones fueran más largas.

6.5. Ampliación del conjunto de datos mediante datos sintéticos

Los modelos genéricos presentan una disminución del rendimiento al evaluarse en un conjunto de datos ruidosos en lugar de uno de alta calidad, como CONLL. Aunque ajustar el modelo con datos del dominio mejora la calidad del etiquetado, la cantidad de ejemplos anotados disponibles es muy reducida, lo que limita la capacidad de generalización.

Se busca mejorar el rendimiento obtenido en la fase inicial de entrenamiento sin recurrir a una anotación manual. Se decidió implementar una generación automática de ejemplos sintéticos etiquetados en formato BIO mediante el uso de LLMs. El objetivo es producir ejemplos que reflejen el dominio del proyecto sin alejarse demasiado de los datos reales y que aporten combinaciones de entidad-contexto no presentes en el conjunto original.

6.5.1. Herramienta de revisión de ejemplos sintéticos

Para controlar la calidad del etiquetado, desarrollamos una herramienta de revisión que permite visualizar, corregir y almacenar los ejemplos ajustados (Figura 6.4). Esta herramienta no solo fue útil para validar el etiquetado, sino también para identificar patrones de error y ajustar los *prompts* de manera iterativa.

La herramienta recibe un archivo JSON con la estructura `{"ejemplo_i": {"tokens": [...], "labels": [...]}}`, verifica la correspondencia entre *tokens* y etiquetas, y muestra estadísticas globales como el total de ejemplos, el porcentaje de errores (discrepancias entre la cantidad de *tokens* y la de etiquetas en un ejemplo) y el porcentaje de modificaciones.

Cada ejemplo se muestra con las palabras resaltadas según la etiqueta correspondiente. Al hacer clic en la palabra, se puede reasignar su etiqueta; los cambios manuales se marcan con un borde punteado para tener trazabilidad.

Además, es posible generar y almacenar un nuevo JSON con las correcciones aplicadas, para poder ser utilizado en el *pipeline* de entrenamiento.

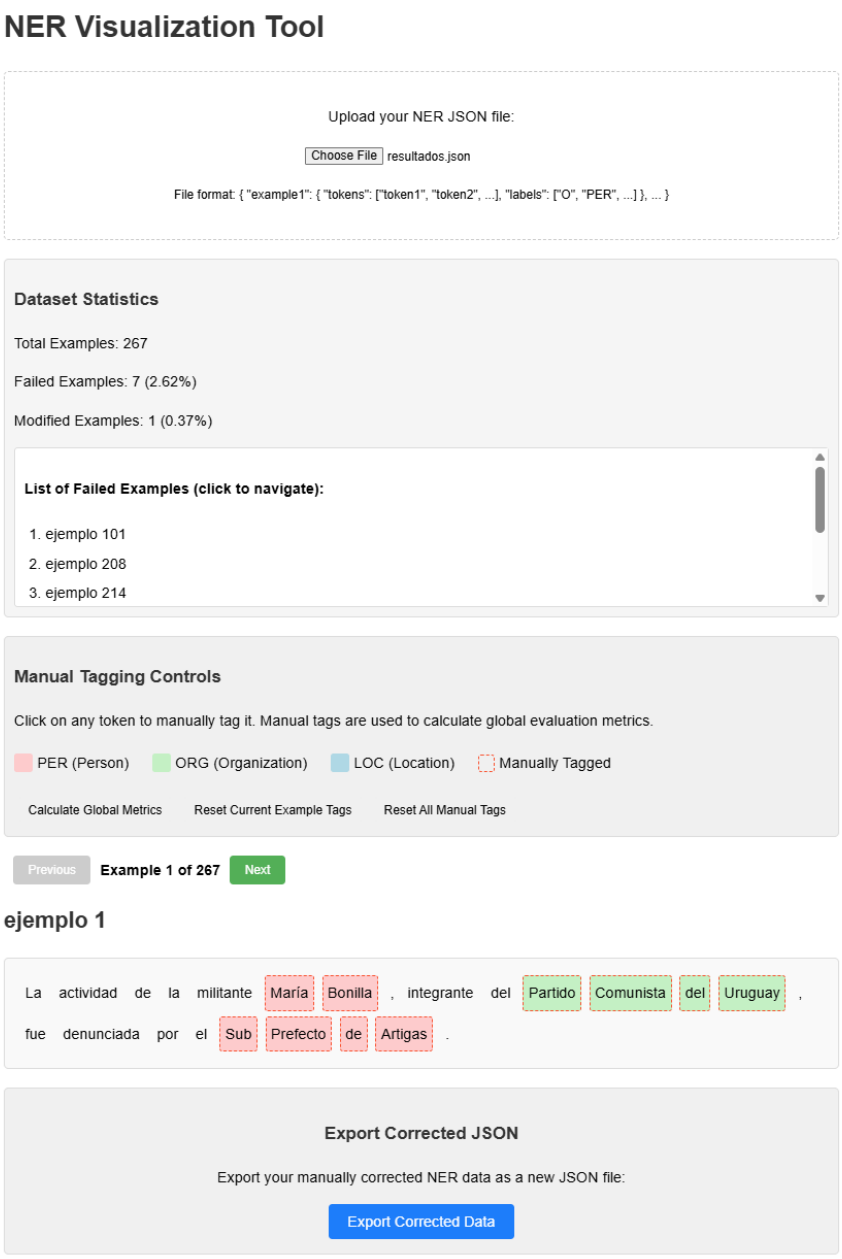


Figura 6.4: Herramienta de visualización de ejemplos NER

6.5.2. Estrategia para la generación de datos sintéticos

Inicialmente, se diseñó un *prompt* (consultar anexo B.1) para instruir al modelo a generar ejemplos sobre textos militares uruguayos entre 1968 y 2004, fechas correspondientes a los archivos originales.

Para asegurarnos de que los ejemplos generados pertenezcan al dominio, se partió de listas predefinidas de organizaciones y de nombres propios obtenidos del proyecto CRUZAR. En cada iteración se seleccionó pseudoaleatoriamente un conjunto de estas entidades, que luego se incluyeron explícitamente en el *prompt*, junto con sus etiquetas correspondientes.

Esta metodología tiene como objetivo evitar que el modelo deba inferir, además de la estructura general de la frase, las entidades y sus etiquetas. Esto permitió generar ejemplos con entidades particulares del dominio que difícilmente habrían sido incluidas por el modelo de lenguaje de forma autónoma. Al agregar las entidades etiquetadas, se reducen las posibilidades de generar ejemplos con errores de etiquetado.

Por otro lado, las entidades de tipo lugar (LOC) fueron tanto generadas como etiquetadas por el propio LLM durante la misma inferencia. Además, el *prompt* especifica el uso de expresiones comunes del dominio, como «detenida en», «a disposición de» o «integrante de», así como el formato de salida requerido: cada palabra, seguida de su etiqueta (PER, ORG, LOC u O), en una línea distinta, separando cada ejemplo con una línea en blanco. Esto facilita el procesamiento posterior para almacenar los ejemplos etiquetados.

Para guiar el formato de salida, se incluyó un ejemplo de cómo realizar el etiquetado siguiendo la técnica de *prompting* COT descrita en la Sección 2.3.1. En particular, se indican al modelo tres pasos para la resolución del problema:

1. Generación del texto base: Se genera una frase a partir de las entidades extraídas de los diccionarios, sin etiquetas.
2. Identificación de entidades: Se asigna una etiqueta a cada palabra que conforma la entidad.
3. Etiquetado BIO: Se repite la frase generada en el paso uno, asignando una etiqueta a cada palabra.

6.5.3. Generación con LLAMA 3.1

El primer experimento se realizó utilizando el modelo LLAMA 3.1 (8B) (Meta AI, 2024). La generación se realizó con una temperatura de 0,6 con el fin de balancear la variedad de los ejemplos con la calidad de las etiquetas generadas. Con este primer enfoque se generaron 100 ejemplos sintéticos.

Durante las primeras ejecuciones se intentó generar 100 oraciones en una única ejecución, donde se identificaron dos problemas significativos. Por un lado,

las oraciones generadas presentaban estructuras repetitivas que imitaban la estructura del ejemplo proporcionado. Esta falta de variabilidad añadía datos redundantes que no aportaban información nueva y útil al entrenamiento del modelo. Por otro lado, se detectaron inconsistencias en las etiquetas asignadas, como desplazamientos o etiquetas incorrectas, lo cual refleja la dificultad del modelo para realizar simultáneamente las tareas de generación y etiquetado con precisión. Este comportamiento es esperable ya que se le solicita al modelo que realice una tarea compleja para la cual no está entrenado específicamente.

Con el fin de mejorar la diversidad de los resultados, se decidió reducir la cantidad de oraciones generadas por petición a una. Esta estrategia pretende facilitarle al modelo la generación de ejemplos diversos. Aunque se observó una leve mejora en la calidad, los ejemplos seguían siendo demasiado similares a los iniciales y persistían errores en las etiquetas. Por esta razón, se continuó explorando con un modelo alternativo.

6.5.4. Generación con QWQ 32B

En este enfoque se mantuvo una estrategia similar a la implementada con LLAMA, pero utilizando un modelo razonador (contemporáneo al desarrollo del proyecto) y aumentando la cantidad de ejemplos generados por inferencia a cinco. Se replicó el esquema del *prompt* anterior, pero utilizando el modelo QWQ (32B) (Qwen Team, 2025). La temperatura utilizada fue 1, dado que no se observó deterioro en el etiquetado al mantener una temperatura alta.

Los primeros 100 ejemplos generados mostraron una calidad superior a la obtenida con LLAMA; los textos presentan una mayor diversidad y coherencia global. Del total de ejemplos, se alcanzó una precisión del 96 % de ejemplos bien formados (la cantidad de etiquetas igual a la cantidad de palabras); esto se midió automáticamente. Además, dentro de los ejemplos bien formados, manualmente se revisó el etiquetado individual, obteniendo una precisión del 99 %. Los ejemplos con errores de etiquetado fueron corregidos para su uso posterior.

El modelo QWQ, al disponer de un número significativamente mayor de parámetros que LLAMA 3.1, implica un costo computacional mayor en la generación. Sin embargo, dado que este proceso de generación debe realizarse una sola vez, es posible priorizar la calidad de los datos sintéticos por encima del tiempo de procesamiento.

6.5.5. Simulación de ruido OCR y LUISA

Tanto los datos reales como los ejemplos utilizados para evaluar el modelo provienen de documentos transcritos de calidad variable. En estos documentos se encuentran dos fuentes principales de ruido: errores propios de la transcripción OCR (por ejemplo, fragmentación de palabras y sustituciones de caracteres por otros similares) y convenciones y artefactos propios de la plataforma LUISA (por ejemplo, el uso del símbolo «@» como sustituto de palabras ilegibles), que a su

vez pueden incluir segmentaciones y sustituciones análogas a las de OCR.

Para adaptar al modelo a este contexto ruidoso y mejorar su capacidad de generalización, se desarrolló un *script* de preprocesamiento para los ejemplos sintéticos que simula ruido de forma controlada. El diseño es por capas, donde la primera modela siempre el ruido OCR, y sobre ella puede activarse opcionalmente una capa adicional que modela el ruido LUISA.

Las transformaciones implementadas son:

- **Reemplazo de caracteres:** usa un diccionario configurable (consultar Sección C.1) para sustituir letras por otras similares, como la «e» por la «c», la «l» por el «1», la «o» por el «0». Estas sustituciones imitan confusiones similares producidas en la etapa de OCR de documentos deteriorados (ruido OCR) o características presentes en los documentos obtenidos por LUISA (ruido LUISA). El diccionario utilizado puede ser consultado en Sección C.1.
- **Fragmentación de palabras:** con una probabilidad variable, una palabra puede dividirse en dos partes y conservar correctamente sus etiquetas BIO. Esta funcionalidad simula errores de segmentación donde una misma entidad queda dividida por espacios incorrectos.
- **Marcadores de ilegibilidad:** inserción de símbolos característicos de LUISA para señalar tramos no legibles.
- **Inserción de funciones de ruido adicionales:** la estructura del programa permite aplicar transformaciones adicionales según una palabra clave. Esto permite agregar de forma sencilla nuevas funciones de ruido que se adapten a distintas herramientas.

En la Figura 6.5 se muestra un ejemplo de una oración y la misma después de aplicar ruido OCR mediante sustituciones de caracteres y fragmentación.

Ejemplo sintético con y sin ruido

El teniente Sosa, integrante de la Asesoría Letrada de la Armada, fue asignado al Área naval Cerro en 1975.

El temi3nte So$@, in+egrante de ia @s€5 oría Le+r ad@ de 1a Arrnada, fv3 así9nado al Área @ Cerro 3n 1975.

Leyenda de anotaciones: Persona Organización

Figura 6.5: Ejemplo sintético con y sin ruido

En general, la presencia de ruido en el texto hace que cada palabra sea divi-

dida en un mayor número de *tokens*, lo que dificulta resolver el problema de etiquetado (Matthews y collaborators, 2024).

Es posible configurar los parámetros que determinan la probabilidad de realizar cada una de las modificaciones para generar un conjunto de datos sintéticos ruidosos. El etiquetado resultante se ajusta para mantener la coherencia con el esquema BIO cuando las palabras se alteran.

6.5.6. Evaluación de datos sintéticos

La calidad de los datos sintéticos generados se evalúa mediante la evaluación intrínseca y la extrínseca.

Evaluación intrínseca

La calidad de los datos generados puede evaluarse a partir de diversas dimensiones, entre las que se incluyen: la corrección de las etiquetas, la diversidad semántica y la similitud con los datos reales.

En primer lugar, existen casos donde la cantidad de etiquetas generadas no coincide con la cantidad de palabras del ejemplo sintético. Durante la etapa de experimentación, se calculó el número de ocurrencias de esta discrepancia.

Además, se realizó una evaluación manual para comprobar la corrección de las etiquetas generadas, validando que se representen correctamente las entidades. Para realizar las correcciones, se utilizó la herramienta de etiquetado desarrollada.

Para evaluar dimensiones más abstractas de los datos sintéticos, se aprovecha la capacidad de los modelos de tipo *encoder*, como BERT, para obtener representaciones contextuales (*embeddings*) de secuencias de *tokens*. Como se describe en la Subsección 2.3.2, estos modelos pueden entrenarse para generar un *token* especial CLS al inicio de cada secuencia, cuya representación final se utiliza como *embedding* representante de toda la frase. Posteriormente, se aplica el análisis de componentes principales (PCA) para reducir la dimensionalidad de los *embeddings*, lo que facilita la visualización y el análisis comparativo en un espacio de menor dimensión.

Durante la experimentación, se generan *embeddings* tanto para las secuencias generadas como para las reales pertenecientes al conjunto de validación. Esto permite calcular la similitud global entre ambos conjuntos, definiendo el centroide de cada uno como el promedio de sus puntos, proporcionando una medida de qué tan representativos son los datos sintéticos respecto a los reales.

Además, es posible calcular la distancia promedio de cada *embedding* respecto al *centroide* de su propio conjunto, para obtener un valor que puede ser interpretado como indicador de la diversidad semántica del conjunto de ejemplos.

Evaluación extrínseca

Es posible realizar una evaluación extrínseca de la calidad de los datos generados; para ello se utilizan los ejemplos generados para el entrenamiento de uno o varios modelos con el fin de resolver una o varias tareas de interés. A partir de esto, se determina la calidad de los datos sintéticos en función de su efectividad al ser utilizados para resolver dichas tareas.

Si bien este método permite comparar fácilmente la utilidad que los ejemplos generados proveen a la hora de resolver las tareas evaluadas, esta metodología requiere realizar múltiples entrenamientos, por lo que requiere una mayor cantidad de cómputo e impide evaluar la calidad de los datos *a priori*.

6.5.7. Calidad de datos sintéticos

Para validar la utilidad y la representatividad de los datos sintéticos con respecto a los reales, se analizó la cercanía entre los ejemplos y la diversidad interna de cada conjunto en un espacio vectorial común de *embeddings* contextuales generado con un modelo BERT. Se proyectan tanto ejemplos reales como sintéticos en el mismo espacio, lo que permite medir distancias y similitudes.

La distancia entre el *centroide* de los datos reales y el resto de los conjuntos sirve para estimar el grado de similitud global entre ellos. Los resultados que se muestran en la Tabla 6.3.

Las distancias entre el *centroide* de los datos reales y los sintéticos generados con QWQ son de 0,48 (sin ruido) y 0,49 (con ruido), inferiores a CONLL-2002 y mucho menores que LLAMA (2,20). Esto indica que QWQ generó ejemplos más alineados con la distribución de los datos reales que LLAMA. La diferencia entre 0,48 y 0,49 es menor, por lo que no es suficiente para determinar si el ruido será de utilidad o no durante el entrenamiento *a priori*.

Comparación	**Distancia absoluta**
Datos reales vs. Sintéticos: QWQ + Ruido	0,49
Datos reales vs. Sintéticos: QWQ	0,48
Datos reales vs. Sintéticos: LLAMA	2,20
Datos reales vs. CONLL-2002	0,59

Tabla 6.3: Distancia entre *centroides* de los conjuntos evaluados

Por otro lado, para evaluar la diversidad interna de cada conjunto, se calculó la distancia promedio de cada *embedding* a su propio centroide; en la Tabla 6.4 se muestran los valores calculados. Los datos reales muestran una mayor dispersión, mientras que los datos sintéticos de QWQ presentan menor diversidad.

Este comportamiento es esperable dado el uso de diccionarios y un *prompt* que restringe la generación. Sin embargo, una mayor dispersión no indica necesariamente mayor realismo; los sintéticos de LLAMA muestran una alta variabilidad

sustancialmente mayor que la demostrada por los datos reales. Pese a esto, se observó que los ejemplos generados por este modelo siguen una estructura más rígida.

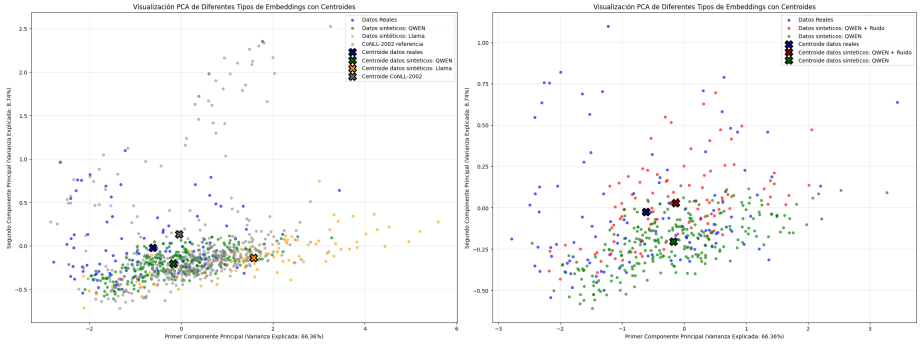
No se incluye la representación de LLAMA con datos ruidosos porque su desplazamiento de distribución con respecto a los datos reales es marcadamente mayor.

En la Figura 6.6a se muestran las proyecciones PCA de los distintos conjuntos y sus *centroides*. Los conjuntos generados con QWQ son más cercanos al *centroide* de los datos reales, mientras que aquellos obtenidos con LLAMA presentan una distancia sustancialmente mayor. Se complementó con la Figura 6.6b, incluyendo la representación de los datos sintéticos de QWQ con ruido, sin tomar en cuenta CONLL-2002 o LLAMA para mejorar la observabilidad.

Conjunto	**Distancia promedio al centroide**
Datos reales	1,09
Sintéticos: QWEN + Ruido	0,80
Sintéticos: QWEN	0,78
Sintéticos: Llama	1,38
CONLL-2002	1,31

Tabla 6.4: Distancia promedio de los *embeddings* al centroide propio

Dado el mejor alineamiento de QWQ y una diversidad compatible con los datos reales, se decidió continuar con este enfoque y descartar los ejemplos generados con LLAMA. En total, se generaron 268 ejemplos sintéticos, que conformaron el conjunto utilizado en experimentos posteriores.



(a) Embeddings: datos sintéticos (b) Embeddings: datos sintéticos con ruido

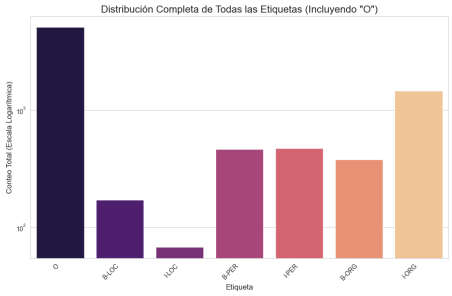
Figura 6.6: Comparación de los *embeddings* contextuales en diferentes condiciones

6.5.8. Distribución de etiquetas en datos sintéticos

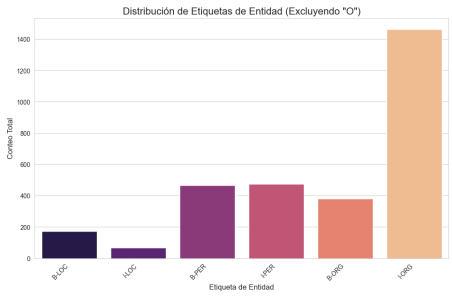
Para evaluar el balance y la cobertura de las entidades en los datos sintéticos generados, se analiza la distribución de etiquetas y su coocurrencia. Esto permite identificar posibles sesgos en las etiquetas y la composición de los ejemplos, lo cual podría afectar el desempeño del modelo.

En la Figura 6.7a se observa que, en sintonía con lo encontrado en los datos reales, el número de etiquetas «O» es mucho mayor que el de apariciones del resto de las etiquetas. Pese a esto, las apariciones de entidades son más frecuentes; se observa que en la totalidad de los ejemplos generados siempre contienen al menos una entidad (Figura 6.7d). Se espera que esta mayor densidad de entidades facilite el aumento del *recall* durante el entrenamiento.

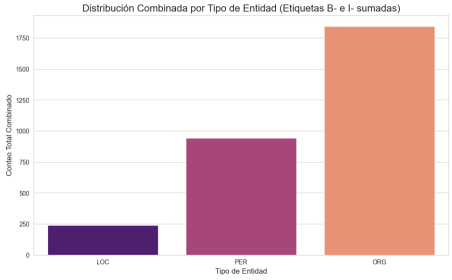
Las entidades presentes en la lista de organizaciones tienen una mayor cantidad de *tokens* que las que se encuentran en los datos reales; esto se ve reflejado en la Figura 6.7c.



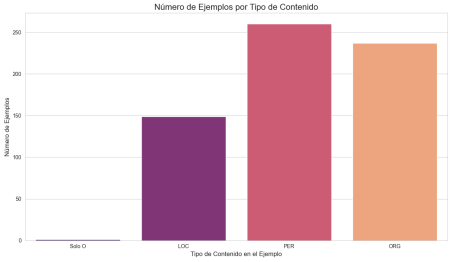
(a) Distribución Completa de Todas las Etiquetas (Sintéticas).



(b) Distribución de Etiquetas de Entidad (Sintéticas).



(c) Distribución Combinada por Tipo de Entidad (Sintéticos).



(d) Número de Ejemplos por Tipo de Contenido (Sintéticos).

Figura 6.7: Distribución de etiquetas y composición de ejemplos en el conjunto de datos sintéticos.

En comparación con los datos reales, el conjunto de datos sintéticos posee una mayor cantidad de ejemplos que contienen entidades de distintos tipos de forma simultánea (Figura 6.8).

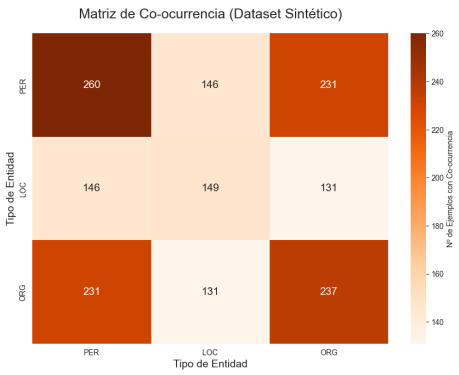


Figura 6.8: Matriz de coocurrencias datos sintéticos etiquetados

6.6. Entrenamiento con datos sintéticos

Una vez obtenido el conjunto de datos sintéticos, se incorpora al entrenamiento de los modelos. En esta sección, se detalla este proceso definiendo nuevos hiperparámetros, estrategias de entrenamiento y analizando los resultados.

Se optimizó la construcción del conjunto de entrenamiento con *Optuna*, incluyendo hiperparámetros asociados a la selección de ejemplos sintéticos. Se ajusta el estilo de ruido para elegir entre agregar solo ruido OCR o añadir además ruido LUISA. También se optimizan la probabilidad de agregar el tipo de ruido seleccionado, la probabilidad de división de palabras y la proporción de datos sintéticos, definida como el porcentaje agregado respecto a la cantidad de datos reales. Por ejemplo, con una proporción del 50 %, si fueran 100 ejemplos reales, se añadirían 50 sintéticos y el conjunto resultante contendría 150 elementos. En todos los casos se utiliza el 100 % de los datos reales disponibles, dado su bajo volumen, con el objetivo de aprovecharlos por completo.

La generación de datos sintéticos se realiza en lotes, donde los ejemplos contenidos en cada uno pueden tener similitud entre ellos, por ejemplo, reutilizando una misma entidad en contextos distintos. Para evitar sesgos por el orden de generación y asegurar resultados reproducibles, el conjunto de datos sintéticos se mezcla de forma determinista para luego ser recortado según la cantidad de ejemplos sintéticos seleccionados por *Optuna*. Esto garantiza una inclusión

creciente; al aumentar la proporción de datos sintéticos, el conjunto de mayor proporción contiene todos los ejemplos ya seleccionados en proporciones menores.

En consecuencia, los ejemplos que quedan al final solo se incorporan cuando se usan proporciones muy altas de datos sintéticos. Esto no implica una pérdida en calidad, ya que se asume homogeneidad al realizar una mezcla determinista de los ejemplos, previa al entrenamiento. Además, los datos se generan con el mismo *prompt* y modelo.

En esta etapa no se presentan los *slice plots* ni el análisis de *learning rate*, *batch size* o *head-only*, porque su comportamiento replica lo observado en el entrenamiento con datos sintéticos.

6.6.1. Entrenamiento secuencial

Como se explicó en la Subsección 2.8.3, el método de entrenamiento secuencial (SAT) consta de dos etapas. En primer lugar, se ajusta el modelo únicamente con datos sintéticos y luego se realiza otro entrenamiento sobre el modelo resultante, empleando los datos reales.

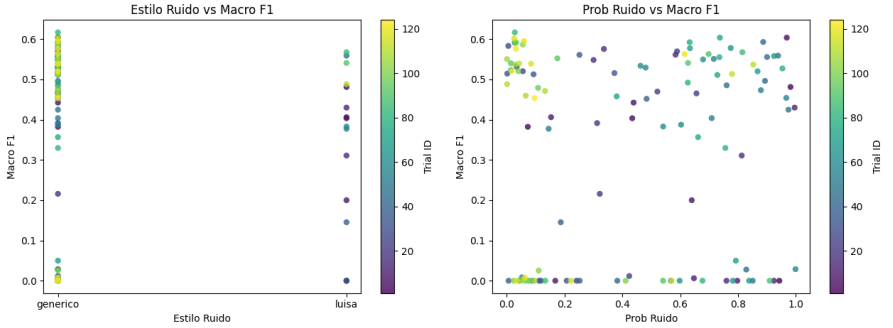
La fase de entrenamiento con datos sintéticos ayuda al modelo a cubrir ejemplos con entidades del dominio, mientras que la fase de entrenamiento con datos reales corrige el desajuste de distribución. La optimización de hiperparámetros es independiente, por lo que el entrenamiento con el conjunto de datos sintéticos y datos reales tiene distintos valores óptimos de *learning rate*, *batch size* o *head only*. Con la mejor configuración de validación, se obtuvo una $F1_{\text{macro}}$ de 0,62.

En la Figura 6.9 se muestra el impacto de los hiperparámetros en el entrenamiento SAT. Se observa que, con el uso de un estilo de ruido genérico, se obtiene la $F1_{\text{macro}}$ más alta. La mayor densidad de puntos con un $F1_{\text{macro}}$ elevado se concentra en este estilo de ruido, mientras que, para el estilo de ruido LUISA, los puntos se distribuyen de manera más uniforme. Esto sugiere que el ruido LUISA no es necesario para que el modelo aprenda a identificar entidades donde uno o varios caracteres son reemplazados por otros.

Si bien la probabilidad de aplicar ruido es baja (0,03), se observa que se alcanzan resultados satisfactorios con una probabilidad más alta. La mayor densidad de puntos asociados a un rendimiento alto se encuentra en las probabilidades más bajas.

Para la probabilidad de división de palabras puede verse que hay una caída en el rendimiento a medida que el porcentaje de división aumenta; con valores elevados, surge un conjunto amplio de ejecuciones con un $F1_{\text{macro}}$ cercano a cero. La causa puede ser estructural; una mayor cantidad de divisiones afecta la tokenización, generando desalineaciones que el modelo no logra componer en el entrenamiento. Por esto, el óptimo quedó en un valor moderado (0,22), suficiente para permitir que el modelo aprendiera a realizar predicciones a pesar de las divisiones presentes en los datos reales.

La proporción de datos sintéticos muestra resultados satisfactorios incluso con valores bajos; hay buenos resultados en un rango amplio, con una pequeña ventaja al incrementar la proporción hasta 1,5 de los datos reales. Más allá de ese umbral, no se muestra ni una mejora sistemática ni colapsos. Esto es coherente con la forma de entrenamiento y el muestreo incremental; al crecer la proporción, solo se agregan ejemplos adicionales sin reemplazar los ya útiles, y la segunda fase con datos reales actúa como filtro para las cosas aprendidas que no tienen correspondencia con el dominio. Esto permite agregar una cantidad de ejemplos sintéticos alta sin perjudicar el rendimiento, aprovechándolos casi por completo.



(a) Estilo ruido vs $F1_{\text{macro}}$

(b) Probabilidad ruido vs $F1_{\text{macro}}$

(c) Probabilidad de división vs $F1_{\text{macro}}$

(d) Proporción de datos sintéticos vs $F1_{\text{macro}}$

Figura 6.9: Resultados de la variación del ruido, la probabilidad de ruido, la probabilidad de división y la proporción de datos sintéticos sobre $F1_{\text{macro}}$.

En resumen, el entrenamiento SAT es una estrategia que permite aprovechar los datos sintéticos sin que estos dominen el proceso de aprendizaje. Como se mencionó en la sección de entrenamiento con datos sintéticos (ver Subsección 2.8.3), la fase inicial con ejemplos sintéticos permite cubrir patrones generales, mientras que la segunda etapa corrige desajustes y alinea el modelo.

Los resultados obtenidos muestran que la introducción de ruido no favorece de manera sustantiva el aprendizaje, mientras que hay una tendencia fuerte hacia una probabilidad de división moderada y una proporción de datos sintéticos alta. Sin embargo, a pesar de alcanzar métricas competitivas con respecto al *baseline*, los resultados no superan el desempeño del entrenamiento realizado únicamente con datos reales, lo que indica que el aprendizaje introducido en el entrenamiento inicial con datos sintéticos no aporta un beneficio sobre el corpus de validación.

6.6.2. Entrenamiento integrado

A diferencia del entrenamiento secuencial, el entrenamiento integrado (IAT) consta de una única etapa en la que se mezclan datos reales y sintéticos.

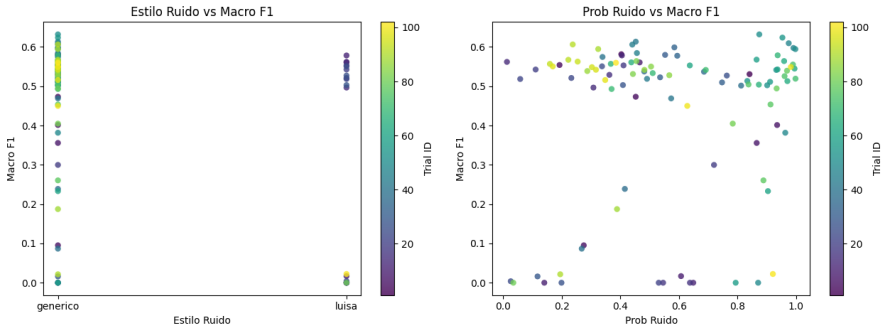
Con la mejor configuración de validación se obtuvo una $F1_{\text{macro}}$ de 0,63, superando los resultados obtenidos con SAT y alcanzando el entrenamiento sin datos sintéticos.

En la Figura 6.10 se observan los *slice plots* de los distintos hiperparámetros estudiados. El estilo de ruido que produjo los mejores resultados fue el genérico, reforzando el resultado obtenido con SAT. El comportamiento es similar, donde la concentración de corridas con $F1_{\text{macro}}$ alto es mayor en genérico y aparecen valores más bajos en el estilo LUISA. Al igual que con el entrenamiento SAT, el ruido genérico aporta suficiente información sin necesitar las modificaciones adicionales para simular los ejemplos en validación.

El valor óptimo de la probabilidad de ruido se encuentra en la franja de valores más altos, alcanzando la mayor $F1_{\text{macro}}$ de 0,87. Esto sugiere que el ruido es útil para que el modelo aprenda a identificar los errores introducidos por la transcripción OCR. Algo similar ocurre con la probabilidad de división, que también se ubicó en valores elevados (0,71), lo que refuerza la idea de que introducir cortes en *tokens* y palabras es útil para simular errores frecuentes en los documentos de entrada.

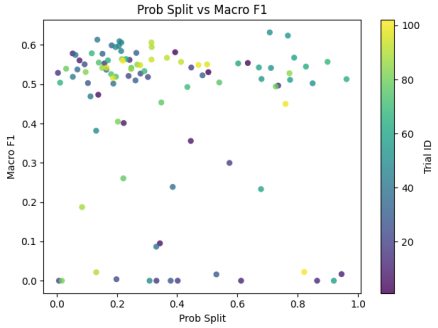
La proporción de datos sintéticos seleccionada por *Optuna* es de 0,5, lo que da como resultado un conjunto con un tercio de datos sintéticos y dos tercios de datos reales. La cantidad de datos sintéticos utilizados es coherente con el tipo de entrenamiento.

En el caso de IAT, donde los datos reales y sintéticos se mezclan en cada *batch*, aumentar excesivamente la proporción de datos sintéticos hace que muchos *batches* estén mayoritariamente conformados por ejemplos artificiales. Esto genera gradientes que orientan al modelo a aprender características de los datos sintéticos en lugar de las reales, lo que termina disminuyendo el rendimiento del modelo en el conjunto de validación. Por otro lado, mantener una proporción moderada de datos sintéticos implica darle mayor importancia a los datos reales, que son los más cercanos a los datos de validación.

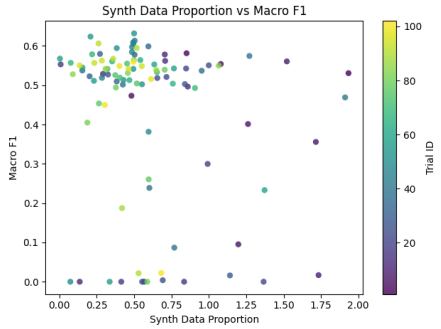


(a) Estilo ruido vs $F1_{\text{macro}}$

(b) Probabilidad ruido vs $F1_{\text{macro}}$



(c) Probabilidad de división vs $F1_{\text{macro}}$



(d) Proporción de datos sintéticos vs $F1_{\text{macro}}$

Figura 6.10: Resultados de la variación del ruido, la probabilidad de ruido, la probabilidad de división y la proporción de datos sintéticos sobre $F1_{\text{macro}}$.

En conclusión, el entrenamiento IAT resultó más efectivo que el SAT. El resultado es coherente con el estudio presentado en la Subsección 2.8.3, que afirma que en escenarios de datos reales escasos (menos de 200 ejemplos) el entrenamiento IAT obtiene mejores resultados que el SAT.

Además, se iguala el puntaje $F1_{\text{macro}}$ obtenido en el entrenamiento, sin datos sintéticos, con el obtenido en la validación.

El uso de ruido genérico, junto con probabilidades elevadas tanto de inserción de ruido como de división de palabras, permitió simular adecuadamente los errores de transcripción presentes en el conjunto de validación. Estos resultados muestran que IAT permite aprovechar los datos sintéticos sin pérdida de rendimiento.

6.6.3. Entrenamiento integrado modificando la métrica de evaluación

Dado que los mejores resultados en validación se obtuvieron utilizando el tipo de entrenamiento IAT en lugar de SAT, seguimos con este modelo para explorar el impacto de modificar la métrica de evaluación empleada durante la búsqueda de hiperparámetros.

Como se explicó en la Subsección 2.5.2, *Optuna* realiza la búsqueda de hiperparámetros generando dos distribuciones de probabilidad: una que modela los conjuntos de hiperparámetros asociados a buenos resultados y otra que modela los asociados a malos resultados. A partir de ellas, sugiere nuevas configuraciones que incrementen la probabilidad de pertenecer al conjunto de buenos resultados y reduzcan la de pertenecer al de malos.

El inconveniente de utilizar esta herramienta en modelos de NER radica en la forma de medir el rendimiento: comúnmente se mide a nivel de entidad. Esto significa que todos los *tokens* de una entidad deben clasificarse con la misma etiqueta para que sea identificada como tal. De lo contrario, se considera falso negativo.

La implicancia de la observación anterior es que las métricas a nivel de entidad no detectan mejoras parciales a nivel de *token* entre dos modelos. Por ejemplo, un modelo que clasifica todos los *tokens* como «O» tiene la misma $F1_{\text{macro}}$ que un modelo que detecta el 90 % de los *tokens* de todas las entidades.

Esto lleva a que *Optuna* clasifique como mala una configuración que no logra mejorar significativamente la detección completa de entidades, pero muestra una mejora en la clasificación de *tokens* individuales. Aunque estas mejoras parciales no son suficientes por sí solas para lograr un mejor desempeño global (ya que el objetivo es reconocer la entidad completa), ignorarlas dificulta la búsqueda al descartar combinaciones de hiperparámetros que podrían conducir a un mejor modelo tras varias ejecuciones.

Por esta razón definimos una métrica propia que refleje no solo el desempeño a nivel de entidad, sino también las mejoras a nivel de *token*. La métrica se define como la medida armónica entre la $F1_{\text{macro}}$ a nivel de entidad y la $F1_{\text{macro}}$ a nivel de *token*. De esta forma, se incentiva al modelo a identificar correctamente entidades completas, sin descartar pequeños avances en la clasificación de *tokens* que podrían traducirse en un mejor desempeño global.

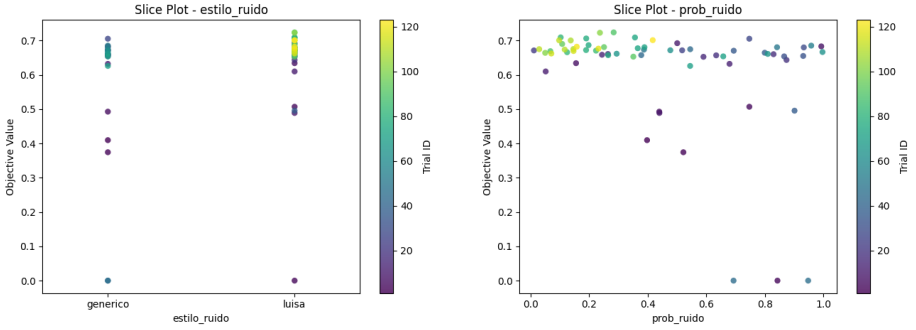
$$F1_{\text{combinado}} = \frac{2 \cdot F1_{\text{entidad}} \cdot F1_{\text{token}}}{F1_{\text{entidad}} + F1_{\text{token}}}$$

$$F1_{\text{combinado}} = \frac{4 \cdot P_{\text{entidad}} \cdot R_{\text{entidad}} \cdot P_{\text{token}} \cdot R_{\text{token}}}{(P_{\text{entidad}} + R_{\text{entidad}})\, P_{\text{token}}\, R_{\text{token}} + (P_{\text{token}} + R_{\text{token}})\, P_{\text{entidad}}\, R_{\text{entidad}}}$$

Para la búsqueda de hiperparámetros, se modificó el objetivo de *Optuna* por

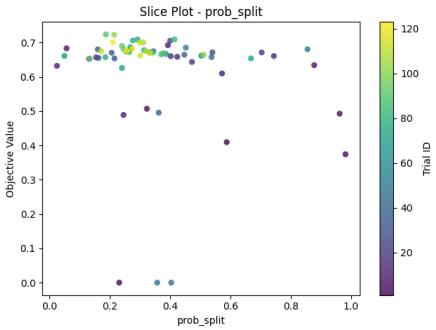
el $F1_{\text{combinado}}$; el cambio busca que no se pierdan mejoras parciales. Con esta métrica, IAT logra un nuevo mejor resultado en validación: $F1_{\text{macro}} = 0{,}65$, por encima del mejor valor previo de 0,63, logrando un incremento de 2 puntos porcentuales.

Para el estilo de ruido, el preferido por *Optuna* es el ruido LUISA; sin embargo, se observa en la Figura 6.11 que la diferencia entre el mejor valor obtenido para la $F1_{\text{macro}}$ combinada entre ruido genérico y LUISA es muy baja. Esto sugiere que el estilo de ruido aplicado no tiene un impacto significativo en el rendimiento del modelo.

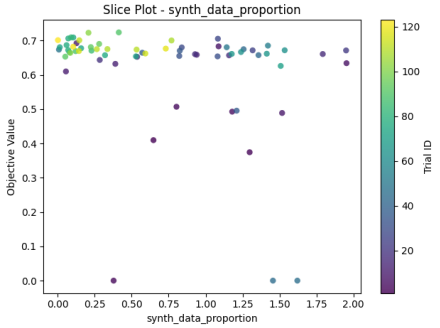


(a) Estilo de ruido vs $F1_{\text{macro}}$

(b) Probabilidad de ruido vs $F1_{\text{macro}}$



(c) Probabilidad de división vs $F1_{\text{macro}}$



(d) Proporción de datos sintéticos vs $F1_{\text{macro}}$

Figura 6.11: Resultados de variación de estilo de ruido, probabilidad de ruido, probabilidad de división y proporción de datos sintéticos sobre $F1_{\text{macro}}$.

Para la probabilidad de ruido se observan valores relativamente acotados; el mejor *trial* es 0,28, dentro de una zona estable.

En la probabilidad de división existe un decremento del rendimiento en valores medios y altos, que podrían estar relacionados con desalineaciones BIO por la fragmentación de *tokens*. El valor óptimo encontrado por *Optuna* es 0,19; sufi-

ciente para que el modelo aprenda a manejar este tipo de errores en el conjunto de validación sin generar demasiados errores de tokenización. Estos errores se deben a que las palabras con ruido se tokenizan de forma distinta a las sin ruido.

La proporción de datos sintéticos seleccionada es 0,41 y el análisis es muy similar al realizado para IAT donde el objetivo de *Optuna* es la $F1_{\text{macro}}$ a nivel de entidad.

El cambio de objetivo en *Optuna* a $F1_{\text{combinado}}$ mejoró el desempeño en el corpus de validación a nivel de entidad. La nueva métrica reorienta al modelo hacia configuraciones que convierten mejoras parciales de *token* en ganancias a nivel de entidad.

Síntesis de resultados en validación

A modo de cierre de esta sección, en la Tabla 6.5 se resumen las métricas de validación de cada variante.

Tabla 6.5: Resultados de desempeño en **validación** para SAT, IAT, IAT mejorado, sin datos sintéticos y *Stanza CoNLL-2002* (*baseline*).

Métrica	**SAT**	**IAT**	**IAT Mejorado**	**Sin sintéticos**	***Stanza CoNLL-2002***
F1 Macro	0,62	0,63	**0.65**	0,63	0,28
Accuracy	**0.95**	**0.95**	**0.95**	0,94	0,87
Precision PER	0,69	**0.74**	0,65	0,62	0,48
Recall PER	0,57	0,57	**0.72**	0,57	0,44
F1 PER	0,63	0,65	**0.68**	0,60	0,46
Precision ORG	0.58	**0.73**	0,62	0,51	0,36
Recall ORG	**0.76**	0,71	0,71	0,74	0,38
F1 ORG	0,66	**0.72**	0,66	0,60	0,37
Precision LOC	0,52	0,57	**0.72**	0,67	0,29
Recall LOC	0,63	0,50	0,54	**0.75**	0,29
F1 LOC	0,57	0,53	0,62	**0.71**	0,29

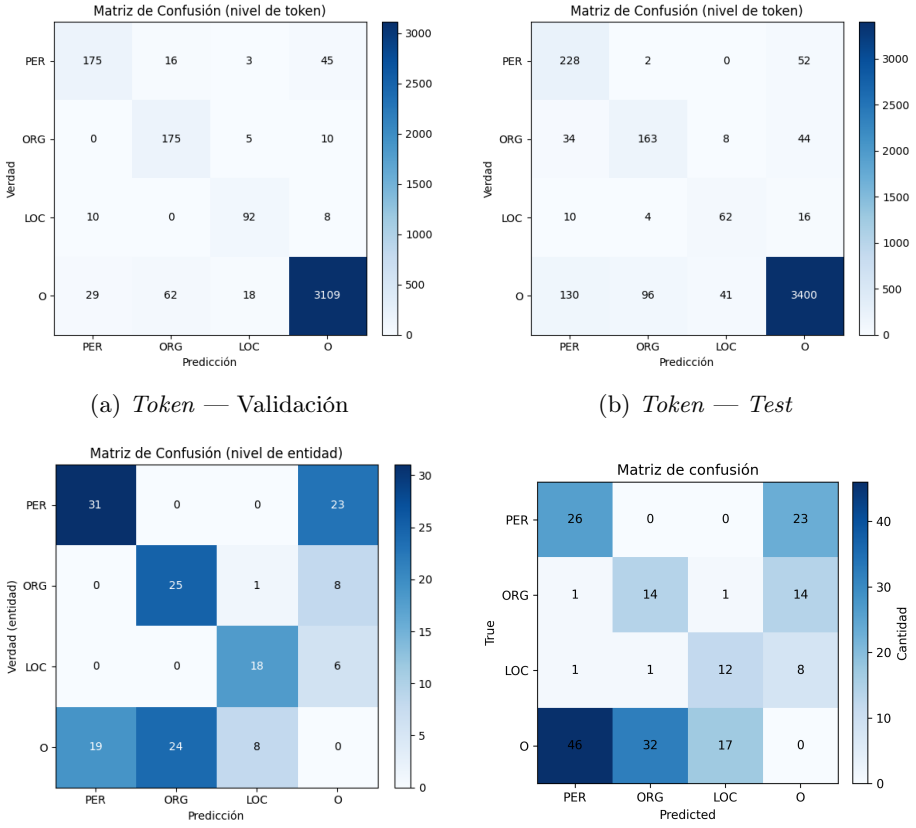
6.7. Evaluación comparativa en *test*

Partiendo de los resultados de validación sintetizados en la Tabla 6.5, se evalúa el rendimiento de las distintas variantes en el conjunto de *test* y se compara con el *baseline* obtenido con *Stanza*. La comparación considera tanto el impacto de incorporar datos sintéticos como el efecto del ruido y las diferentes configuraciones de entrenamiento. Además, se analiza el impacto de cada modelo en los resultados a nivel de *token* y de entidad. De esta forma, se busca identificar no solo qué configuración alcanza las mejores métricas, sino también qué tipo de errores predominan en cada caso y cómo se distribuyen entre las clases PER, ORG y LOC.

6.7.1. Modelo entrenado sin datos sintéticos

Para el entrenamiento sin datos sintéticos, el rendimiento en el conjunto de *test* desciende con respecto al de validación (Tabla 6.7). La pérdida de $F1_{\text{macro}}$ es de 0,21. Por clase, la métrica $F1_{\text{macro}}$ desciende de 0,60 a 0,42 en PER, de 0,60 a 0,34 en ORG y de 0,71 a 0,46 en LOC.

Las matrices de confusión a nivel de *token* (Figura 6.12) nos permiten deducir la razón principal del decremento del rendimiento; hay un aumento en la cantidad de *tokens* de entidades reales que son etiquetadas como O. Para ORG, la proporción en la que se confunde con O aumenta de manera más acentuada (de 0,05 a 0,18), y para LOC también aumenta (de 0,07 a 0,17). Por otro lado, para PER se reduce levemente, quedando prácticamente igual. Este comportamiento sugiere que, al enfrentarse a ejemplos nuevos, el modelo es más conservador al identificar una entidad, sobre todo para ORG y LOC.



(a) *Token* — Validación

(b) *Token* — *Test*

(c) Entidad — Validación

(d) Entidad — *Test*

Figura 6.12: Matrices de confusión para validación y *test sin sintéticos*: arriba, a nivel de *token*; abajo, a nivel de *entidad*.

A nivel de entidad (Figura 6.12) la disminución del rendimiento en ORG y LOC aumenta como consecuencia de errores de etiquetado de entidad completa. En *test*, la etiqueta ORG presenta el doble de confusión con la etiqueta O que en validación. En LOC la pérdida es muy similar. Esto significa que las confusiones con la entidad LOC se deben a no identificarla, mientras que para ORG se deben a no identificarla completamente.

Esto concuerda con la naturaleza del corpus; las entidades LOC suelen tener una menor cantidad de *tokens* que ORG; mientras que ORG tiene 3,1 *tokens* en promedio por entidad, LOC tiene 2,5 (Subsección 6.1.1).

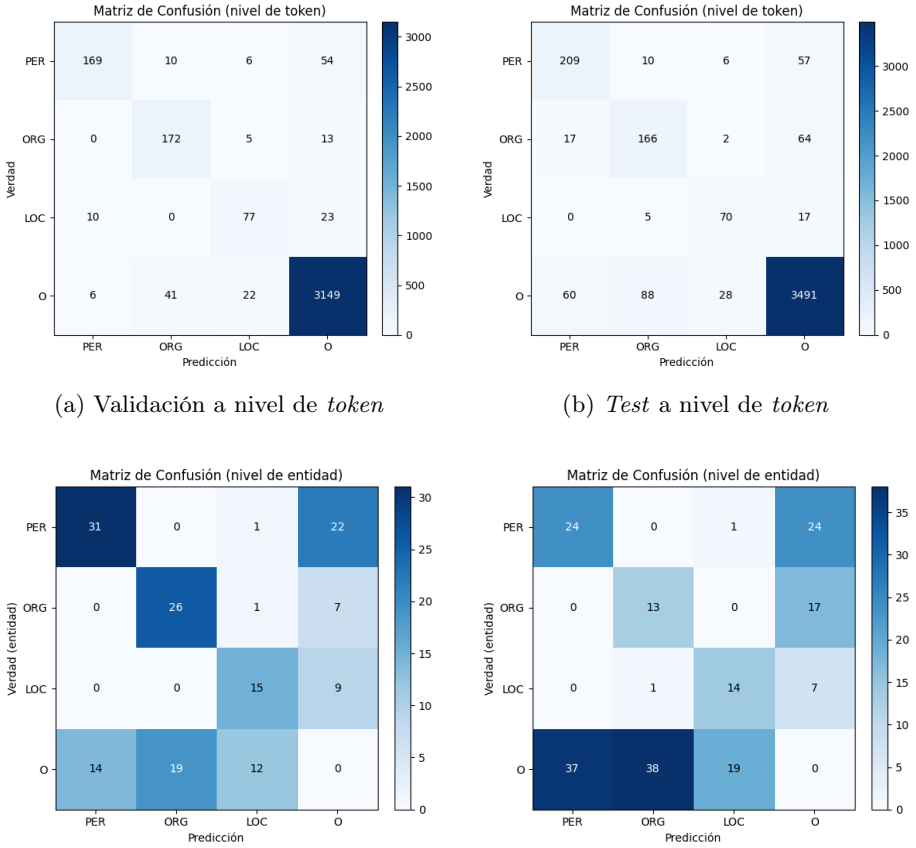
El contraste entre validación y *test* sugiere que la cantidad de ejemplos disponibles no es suficiente para ser representativa, ya que contiene información no representada en los conjuntos de entrenamiento y validación. Las organizaciones son entidades compuestas, por lo que son más sensibles a variaciones en el límite de la entidad o al no etiquetado por parte de la entidad, lo que provoca una confusión con O. En LOC ocurre algo similar pero menos pronunciado; las direcciones compuestas pueden provocar la aparición de errores del mismo tipo. Para PER, la mejora en *recall* implica que los nombres se conservan mejor a pesar del ruido, pero la caída en la precisión sugiere que el modelo extiende el etiquetado a *tokens* adyacentes.

En síntesis, el error dominante es la confusión con la clase O y la desalineación de la detección de entidades. De todas formas, el modelo supera los resultados del *baseline*.

6.7.2. Modelo entrenado con datos sintéticos

Entrenamiento secuencial

Para el entrenamiento secuencial con datos sintéticos, se realiza un análisis a partir de las matrices de confusión presentadas en la Figura 6.15. Se puede observar que, en validación, a nivel de *token*, la mayor parte de los errores son por confusiones con la clase O. Al pasar al conjunto de *test*, aumentan los falsos positivos debido a este tipo de errores. Esto implica una caída de precisión a nivel de entidad. Por ejemplo, en validación hay 6 *tokens* que deberían ser clasificados como clase O, pero en su lugar son clasificados como PER, mientras que en *test* el número aumenta a 60; para ORG el cambio es de 41 a 88 y para LOC de 22 a 28.



(a) Validación a nivel de *token*

(b) *Test* a nivel de *token*

(c) Validación a nivel de *entidad*

(d) *Test* a nivel de *entidad*

Figura 6.13: Matrices de confusión para el conjunto SAT en validación y *test*, a nivel de *token* y entidad.

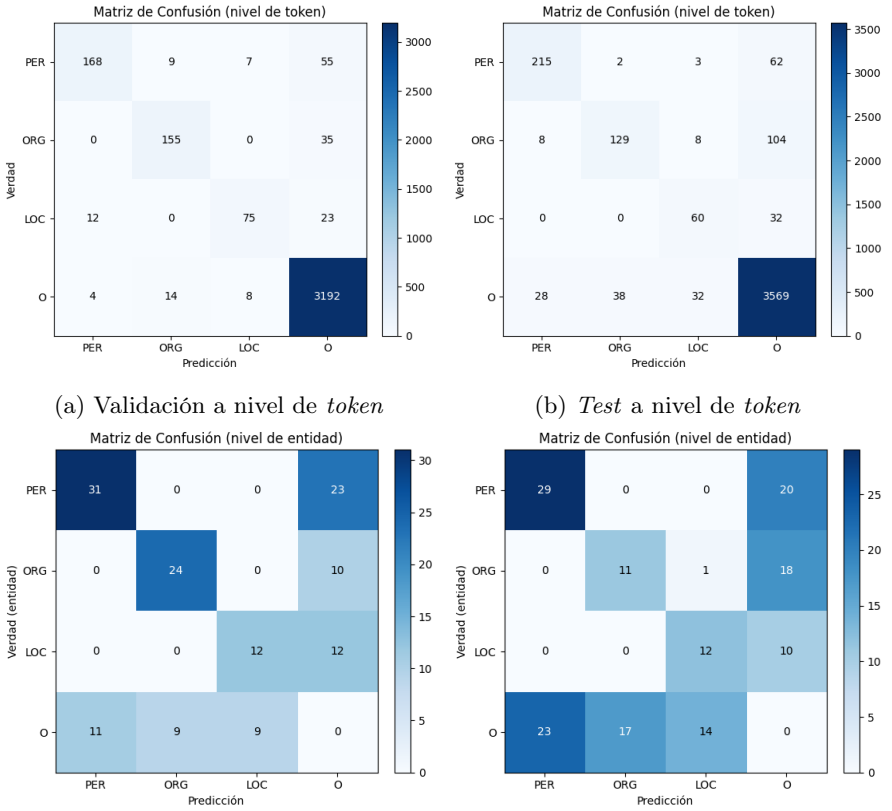
Para las matrices a nivel de entidad, la confusión entre las clases PER, ORG y LOC es mínima. Cuando el modelo identifica una entidad, suele detectar correctamente su tipo.

La precisión baja en *test* por dos razones: se identifican como parte de una entidad zonas con únicamente O y se pierden entidades completas por errores de delimitación. Como resultado, hay una notable diferencia entre validación y *test* en la $F1_{\text{macro}}$ a nivel de entidad, con un mayor deterioro en ORG, mientras que LOC conserva mejor el *recall*, agregando falsos positivos en el proceso.

En conclusión, las figuras muestran que el cuello de botella de SAT no se encuentra en la distinción entre las entidades PER, ORG o LOC, sino en la decisión entre entidad y no entidad, y en la segmentación. Estos errores tienen un gran impacto en los resultados a nivel de *entidad*, ya que un único *token* mal clasificado invalida la identificación completa de una entidad.

Entrenamiento integrado

Para el entrenamiento integrado con datos sintéticos, se realiza un análisis a partir de las matrices de confusión (Figura 6.14), que muestra un patrón similar al de SAT. Cuando el modelo decide que hay una entidad, pocas veces se confunde con su tipo; la mayor parte de los errores son introducidos al identificar si un *token* es o no una entidad y en la segmentación de entidades. El rendimiento entre IAT y SAT es similar, aunque IAT reduce la cantidad de etiquetas O que clasifica erróneamente como entidades y, a su vez, reduce levemente la cantidad de confusiones entre entidades.



(a) Validación a nivel de *token*

(b) *Test* a nivel de *token*

(c) Validación a nivel de *entidad*

(d) *Test* a nivel de *entidad*

Figura 6.14: Matrices de confusión del modelo IAT para los datos de validación y de *test*, tanto a nivel de *token* como de *entidad*.

En validación, a nivel de *token*, se reduce la cantidad de confusiones entre *tokens* de distintas entidades con respecto a test, aumentando la precisión con la etiqueta O; esto sugiere que el modelo es más conservador al momento de clasificar un *token* como parte de una entidad.

A nivel de entidad, la observación anterior es más notoria; disminuyen los falsos positivos desde O hacia PER/ORG/LOC. Comparando las tablas 6.5 y 6.7, se puede ver cómo la $F1_{\text{macro}}$ para PER pasa de 0,63 a 0,5; la de ORG fue la que tuvo la mejora más notoria, pasando de 0,66 a 0,72; y, por último, LOC, donde hubo un decremento de 0,57 a 0,53.

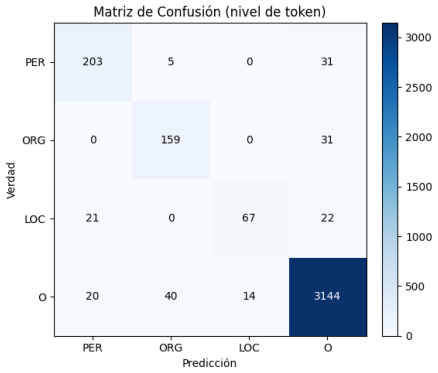
El modelo entrenado con IAT muestra un mejor rendimiento global que SAT y sugiere una mayor capacidad de generalización. Mientras que la diferencia entre la $F1_{\text{macro}}$ en validación y en *test* alcanza un 20 % en SAT, en IAT se reduce a un 15 %, lo que indica una menor caída del desempeño al enfrentarse a ejemplos con los que el modelo no fue ajustado.

Además, IAT es más conservador, con un aumento sistemático de la precisión, consecuencia de la reducción de falsos positivos (principalmente en la clase O, como se muestra en las matrices de confusión tanto a nivel de *token* como de entidad). Esto tiene como consecuencia una disminución del *recall* en ORG y LOC, mientras que en PER el rendimiento se mantiene. En otras palabras, IAT prioriza la corrección de las entidades que predice por encima del número de entidades predichas.

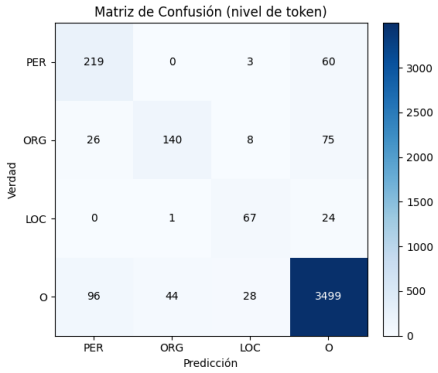
En conclusión, IAT no solo mejora las métricas globales respecto a SAT, sino que también logra una mejor generalización, aunque presenta dificultades en la cobertura de entidades como LOC y ORG.

Entrenamiento IAT modificando la métrica de evaluación

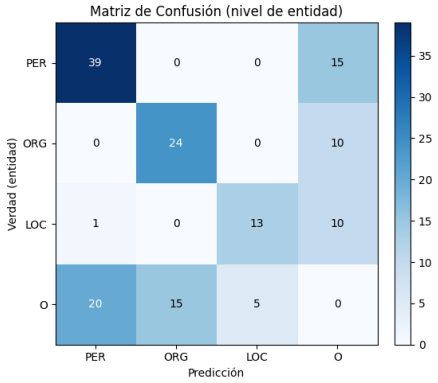
Como ya se mencionó, para esta variante se modificó la métrica de entrenamiento, incorporando en la optimización de *Optuna* la $F1_{\text{macro}}$ a nivel de *token* y de entidad.



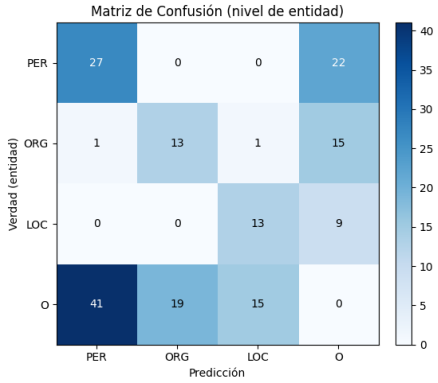
(a) Validación a nivel de *token*



(b) *Test* a nivel de *token*



(c) Validación a nivel de *entidad*



(d) *Test* a nivel de *entidad*

Figura 6.15: Matrices de confusión para el conjunto SAT en validación y *test*, tanto a nivel de *token* como a nivel de entidad.

En validación, los resultados globales son los mejores entre todas las variantes (Tabla 6.5). La $F1_{\text{macro}}$ alcanza 0,65, superando tanto el entrenamiento SAT como el IAT estándar (0,63).

La mejora más notoria es en la clase PER, donde el *recall* aumenta de 0,57 a 0,72, compensando una caída en la precisión. Para la clase ORG, el *recall* se mantiene con una disminución de la precisión, lo que provoca un descenso de la $F1_{\text{macro}}$. Finalmente, para LOC se observa un incremento marcado con respecto al resto

de las variantes que utilizan datos sintéticos, pero aún no alcanza el rendimiento obtenido sin ellos.

Si bien la precisión se ve disminuida, se obtiene una ganancia en cobertura, lo que provoca un aumento de la $F1_{\text{macro}}$. Este comportamiento puede ser beneficioso en el escenario donde se busquen entidades, asumiendo un riesgo de errores en el etiquetado, priorizando una mayor cobertura.

En *test*, el desempeño global es similar al obtenido con IAT estándar, con un $F1_{\text{macro}}$ de 0,46, superando el de SAT. A nivel de clases, mejora el rendimiento en ORG y LOC, disminuyendo considerablemente en PER.

En resumen, utilizar la métrica combinada mejora la cobertura en validación, con un rendimiento inferior al de IAT en *test*, pero superior al de SAT sin datos sintéticos.

Etiquetado por palabra según primer *subtoken*

Como se mencionó en la Subsección 6.2.1, es posible modificar el proceso de inferencia para mitigar errores en el etiquetado de los *tokens* de una palabra, asignando a cada palabra la etiqueta de su primer *token*.

En esta sección se evalúa el rendimiento de todos los modelos anteriores utilizando esta metodología de inferencia.

La metodología de evaluación utilizada en las secciones anteriores requiere que todos los *tokens* correspondientes a una misma palabra compartan la misma etiqueta; de lo contrario, la palabra se asigna a la clase O (ausencia de entidad). Esta forma de inferencia más estricta provoca la pérdida de entidades, ya que un único error en cualquiera de sus *tokens* es suficiente para invalidar la predicción completa de una palabra.

Como resultado, las inconsistencias en el etiquetado de *tokens* pueden afectar negativamente el rendimiento del modelo. Este problema tiene un mayor impacto en el contexto del proyecto, al tratarse de un corpus ruidoso. El *tokenizador* tiende a dividir las palabras en más *tokens* (Matthews y collaborators, 2024), y cuanto más *tokens* tenga, más difícil es que el modelo sea consistente en todos ellos.

En comparación con los resultados sin cambiar la forma de inferencia, la clase PER mantiene su rendimiento con una $F1_{\text{macro}}$ de 0,68, mientras que las clases ORG y LOC mejoran, alcanzando una $F1_{\text{macro}}$ de 0,75 y 0,72, respectivamente.

En *test* (Tabla 6.6), IAT es el que alcanza las mejores métricas globales, con un $F1_{\text{macro}}$ de 0,51, al igual que IAT mejorado, superando al modelo SAT y al entrenado sin datos sintéticos. Todas las variantes mejoran su rendimiento, manteniendo el orden relativo entre ellas.

Tabla 6.6: Resultados de **test** (evaluación a nivel de palabra usando el *primer subtoken*) para SAT, IAT, IAT mejorado y sin sintéticos.

Métrica	SAT	IAT	IAT Mejorado	Sin sintéticos
F1 Macro	0.47	**0.51**	**0.51**	0,48
Accuracy	**0.93**	**0.93**	0,92	0,92
Precision PER	0,45	**0.56**	0,44	0,45
Recall PER	0,51	**0.59**	0,57	**0.59**
F1 PER	0,48	**0.57**	0,50	0,51
Precision ORG	0,28	0,42	**0.46**	0,33
Recall ORG	0,43	0,37	0,43	**0.47**
F1 ORG	0,34	0,39	**0.45**	0,39
Precision LOC	0,50	**0.52**	**0.52**	0,47
Recall LOC	**0.73**	0,59	0,64	0,64
F1 LOC	**0.59**	0,55	0,57	0,54

Evaluar por primer *subtoken* reduce los errores de inconsistencia en el etiquetado de una palabra, mejorando el rendimiento de todas las variantes. IAT con la métrica combinada es la variante preferida por ser la que alcanza los mejores resultados en validación.

Síntesis de resultados en *test*

A modo de cierre de esta sección, en la Tabla 6.7 se resumen las métricas de test para cada variante.

Tabla 6.7: Resultados de desempeño en **test** para SAT, IAT, IAT mejorado, sin datos sintéticos y *Stanza CoNLL-2002* (*baseline*).

Métrica	SAT	IAT	IAT Mejorado	Sin sintéticos	*Stanza CoNLL-2002*
F1 Macro	0,42	**0.48**	0,46	0,42	0,26
Accuracy	0,91	**0.92**	0,91	0,89	0,87
Precision PER	0,39	**0.56**	0,39	0,35	0,27
Recall PER	0,49	**0.59**	0,55	0,53	0,31
F1 PER	0,44	**0.57**	0,46	0,42	0,29
Precision ORG	0,25	0,39	**0.41**	0,30	0,20
Recall ORG	0,43	0,37	0,43	**0.47**	0,30
F1 ORG	0,32	0,38	**0.42**	0,36	0,24
Precision LOC	0,41	0,44	0,45	0,40	**0.46**
Recall LOC	**0.64**	0,55	0,59	0,55	0,55
F1 LOC	0,50	0,49	**0.51**	0,46	0,50

6.7.3. Análisis de resultados y conclusiones

En esta sección se introdujo la generación de datos sintéticos etiquetados como alternativa para mitigar la escasez de datos anotados en un dominio ruidoso.

Los resultados muestran que estos datos sintéticos permiten ajustar la distribución de entrenamiento, incorporando ejemplos cuyo *centroide* se aproxima más al de los datos reales que al de los datos de preentrenamiento.

En todos los casos, los modelos basados en BERT ajustados logran mejoras sustanciales respecto a los obtenidos con STANZA entrenado con CONLL-2002. El uso de datos sintéticos mejora o mantiene el rendimiento en todos los casos. Esto es consistente con la proporción de datos sintéticos seleccionada por *Optuna* durante la optimización de hiperparámetros.

La variante con mejor desempeño fue la que entrena al modelo con IAT, utiliza la métrica combinada como objetivo y realiza la inferencia asignando a cada palabra la etiqueta del primer *token*.

La $F1_{\text{macro}}$ en validación aumenta un 4 % si se considera solo PER y ORG; al incluir LOC, el aumento se reduce a 1 %. En el conjunto de *test*, los mejores modelos, con y sin datos sintéticos, alcanzan $F1_{\text{macro}}$ de $0,51$ y $0,48$, respectivamente.

Este comportamiento indica que los datos sintéticos no mejoran el rendimiento de todas las clases de forma homogénea. Mientras PER y ORG se benefician, la clase LOC se ve perjudicada en todas las variantes que las incorporan.

Esto puede deberse al sesgo introducido en la generación de datos sintéticos. Para generar los ejemplos, se proporcionó al modelo una lista explícita de nombres y organizaciones, asegurando su presencia, la variedad y el etiquetado correcto. En contraste, no se proporcionó una lista predefinida de lugares, dejando que el modelo los generara y los etiquetara. Como resultado, la cobertura de entidades de este tipo fue menor y más ruidosa, lo que dificultó el aprendizaje del modelo NER.

Para mejorar los resultados de forma global, se podría generar una lista de lugares y forzar su uso en la generación de ejemplos, asegurando una representación suficiente de esta clase y una distribución más cercana a la del conjunto de validación.

Considerando la clasificación basada en todos los *tokens* que componen una palabra, el mejor modelo alcanza una $F1_{\text{macro}}$ de $0,48$ en *test*. Este resultado corresponde al entrenamiento IAT y a la $F1_{\text{macro}}$ estándar como métrica de optimización de hiperparámetros. En estas condiciones, se observa un aumento de 6 % al incorporar datos sintéticos en el entrenamiento.

Una parte importante del entrenamiento consiste en el uso de *Optuna* para la búsqueda de hiperparámetros. Si bien esta herramienta permitió explorar de forma eficiente una amplia variedad de configuraciones, optimizar la métrica objetivo sobre el conjunto de validación con una cantidad extensa de pruebas introduce un riesgo de sobreajuste. Las configuraciones seleccionadas pueden terminar capturando particularidades del conjunto de validación en lugar de características generales. Esto explicaría que las mejoras observadas en validación no se reflejan con la misma magnitud en el conjunto de *test*.

En un escenario ideal, con un mayor volumen de datos, el modelo podría generalizar mejor incluso tras una búsqueda extensa de hiperparámetros, ya que los ejemplos de validación serían lo suficientemente diversos como para representar la distribución real del problema. Sin embargo, en contextos con datos limitados, como el presente, se podría mejorar si se evaluara el rendimiento mediante validación cruzada.

Como posible mejora, se propone desacoplar la detección y la clasificación de la entidad en dos etapas: primero, entrenar un modelo que resuelva la decisión binaria entidad vs O, y, solo para los segmentos de texto aceptados, aplicar el etiquetador actual para clasificar el tipo. Esta implementación es coherente con el análisis de las distintas versiones, ya que el cuello de botella principal consiste en detectar la presencia de una entidad. Una vez detectado, el modelo rara vez se confunde con el tipo (PER / ORG / LOC).

En conclusión, los resultados respaldan el uso de datos sintéticos como complemento de los datos reales, siempre que su generación se controle y se evalúe. Como mejora a futuro, se puede equilibrar la representación de LOC en los datos sintéticos corrigiendo el sesgo generado, obtener resultados de validación más representativos utilizando validación cruzada y explorar la detección y clasificación en dos etapas.

Capítulo 7

Conclusiones y Trabajo Futuro

Este proyecto tuvo como objetivo desarrollar un sistema para extraer información de documentos históricos. Para ello, definimos un *pipeline* modular para el procesamiento de texto y la extracción de entidades nombradas utilizando grandes modelos de lenguaje.

Los resultados permiten afirmar que las herramientas desarrolladas superan las iniciativas previas, tanto en la etapa de OCR como en la de NER, aportando mejoras en la calidad de las transcripciones y en el reconocimiento de entidades nombradas, en comparación con las herramientas utilizadas hasta ahora en CRUZAR.

7.1. Resultados principales

En primer lugar, y en línea con lo expuesto en Objetivo Específico II, se implementó un proceso de reconocimiento óptico de caracteres basado en modelos multimodales. La principal dificultad en esta etapa fue la calidad y la diversidad de los documentos: entre los archivos procesados se encuentran fichas, textos manuscritos, mecanografiados y con disposiciones (como columnas), lo que dificultó la transcripción. Simultáneamente, no se dispone de un conjunto de datos con imágenes y transcripciones de alta calidad.

Se realizó un análisis comparativo del rendimiento de múltiples modelos en esta tarea y se exploró el impacto de un paso de preprocesamiento simple en la calidad de las transcripciones y en el tiempo de ejecución. Además, se estudió cómo la calidad de los documentos originales impacta en el proceso de transcripción, lo que permitió determinar bajo qué condiciones el costo computacional propio de los LLMs ofrece el mayor beneficio frente a estrategias tradicionales.

Luego, en línea con el Objetivo Específico V, se implementó un módulo de reconocimiento de entidades nombradas utilizando un modelo basado en BERT entrenado y ajustado al dominio. Dado que la cantidad de datos etiquetados disponibles era muy limitada, se generaron datos sintéticos para ampliar el conjunto de entrenamiento (Objetivo Específico IV). Fueron generados con sus etiquetas, utilizando como referencia diccionarios de organizaciones y nombres, permitiendo que el modelo generara los lugares. Además, se implementó un módulo para simular los artefactos de ruido comúnmente encontrados en textos transcritos mediante OCR. La calidad de los datos generados se evaluó de forma intrínseca para determinar el conjunto de datos sintéticos a utilizar en etapas posteriores.

Con el fin de optimizar el proceso de entrenamiento, se utiliza *Optuna* para ajustar los hiperparámetros. Mediante esta herramienta se determinaron, entre otros parámetros, la cantidad de datos sintéticos a utilizar en el entrenamiento, así como las características del ruido aplicado.

Se propuso una nueva métrica de evaluación con el objetivo de optimizar los hiperparámetros, implementando un criterio que considera simultáneamente el rendimiento del modelo tanto a nivel de entidad como a nivel de *token*.

Se evaluaron múltiples estrategias de entrenamiento (entre ellas SAT e IAT) e inferencia. Se determinó que los mejores modelos son aquellos cuya búsqueda de hiperparámetros se realizó con entrenamiento integrado e inferencia a partir del primer *token*, obteniendo el mismo resultado utilizando la métrica $F1_{\text{macro}}$ como objetivo, como con la métrica que combina tanto la $F1_{\text{macro}}$ a nivel de token como a nivel de entidad. Este modelo alcanzó el mejor rendimiento en el conjunto de *test*. Estos modelos obtienen una $F1_{\text{macro}}$ de 0,51 en el conjunto de *test*, superando al *baseline*, que alcanzó 0,26. Se determina que los datos sintéticos generados permiten mejorar el rendimiento del modelo.

Por último, los objetivos I y III se corresponden con artefactos producidos directamente en el proyecto. El primero corresponde al Capítulo 2 del informe y el segundo corresponde a un repositorio que contiene los conjuntos de datos construidos.

Todo lo expuesto anteriormente permite afirmar que los objetivos del proyecto y el Objetivo General se cumplieron con éxito.

7.2. Mejoras a futuro

7.2.1. Corrección post-OCR

Como línea a futuro, se podría implementar una etapa de corrección post-OCR basada en LLMs restringiendo la decodificación; este procesamiento ya fue introducido en el proyecto «Post-OCR Correction Using Large Language Models with Constrained Decoding» (Sastre y cols., 2025). La implementación presentada ajusta un LLM y, al momento de la inferencia, obliga que la salida mantenga

una alta similitud carácter a carácter con el texto OCR, con el objetivo de reducir alucinaciones y evitar que se introduzca ruido que se traslade a las próximas etapas del *pipeline*. Dado el costo adicional, se puede utilizar el puntaje de legibilidad introducido en el proyecto «*A computational framework for the analysis of the Uruguayan dictatorship archives*» (Etcheverry y cols., 2021) para decidir a partir de qué puntaje resulta beneficioso aplicar el postprocesamiento.

7.2.2. Mejoras a diccionarios de entidades

Una de las principales dificultades identificadas en la generación de datos sintéticos es el sesgo introducido en la generación respecto a los lugares; el método de generación de datos puede mejorarse incorporando una lista de lugares de Uruguay a partir de la información disponible en bases de datos como Wikidata[1]. Esta estrategia podría mejorar el rendimiento global del módulo, mejorando el rendimiento de la clase LOC. En general, incrementar el número de elementos en los diccionarios de referencia utilizados puede permitir aumentar la variedad semántica de los resultados.

También sería posible controlar con mayor precisión la cantidad de entidades de cada clase generadas, con el fin de simular con mayor exactitud la distribución presente en los datos reales o de intentar mejorar el rendimiento en las clases que presentan mayor dificultad.

7.2.3. Mejoras en *prompting*

Otra mejora posible es optimizar el *prompt* explorando variantes de *chain of thought* o mejorando los ejemplos incluidos (*few-shot*). Por ejemplo, es posible variar los datos de referencia cada cierto número de ejemplos sintéticos generados, con el fin de disminuir el riesgo de que el modelo genere datos con estructuras similares.

7.2.4. Métricas, evaluación e inferencia

En el contexto del proyecto, es importante que no se pierdan menciones a entidades, por lo que podría utilizarse una métrica $F1_\beta$ en la que
beta sea mayor que uno, priorizando la *recall* por sobre la precisión. Además, al utilizar una gran cantidad de pruebas *Optuna*, es posible introducir sesgos que favorezcan las características del conjunto de validación. Para mitigar esta problemática, se puede implementar validación cruzada en la evaluación. Si bien este enfoque aumenta el tiempo de ejecución, hace que las métricas sean más representativas del desempeño de los modelos.

Además, durante el desarrollo se emplearon dos métodos de inferencia para la tarea NER (ver Subsección 6.2.1). A futuro, es posible experimentar con métodos más sofisticados. Por ejemplo, seleccionar la clasificación de la etiqueta más común entre los *tokens* de la palabra.

[1] https://www.wikidata.org/wiki/Wikidata:Main_Page

7.2.5. Capacidad de cómputo nacional

Gracias a los esfuerzos nacionales para mejorar la capacidad de cómputo fue posible realizar el procesamiento requerido en este proyecto.

La capacidad de cómputo nacional es un factor necesario para abordar de manera eficiente problemas de alta complejidad y con aplicación directa a la realidad nacional. El uso de LLMs implica un gran costo computacional, por lo que es importante contar con infraestructura y acceso a GPUs.

En algunas ocasiones, la disponibilidad del *Cluster* se veía comprometida por la alta tasa de uso. En esos casos, las tareas como la generación de ejemplos sintéticos y otras tareas se detenían en su totalidad.

Si bien no es una mejora que esté al alcance del equipo, nos parece pertinente anunciar que la inversión en infraestructura es la que posibilita el desarrollo

Referencias

Baek, Y., Lee, B., Han, D., Yun, S., y Lee, H. (2019). Character region awareness for text detection. *arXiv preprint arXiv:1904.01941*. Descargado de https://arxiv.org/abs/1904.01941

Bhadauria, D., Múnera, A. S., y Krestel, R. (2024). The effects of data quality on named entity recognition. *Proceedings of the Ninth Workshop on Noisy and User-generated Text (W-NUT 2024)*, 79–88. Descargado de https://aclanthology.org/2024.wnut-1.8.pdf

Cardozo Ramírez, L., Rivero Cor, L., y Zorrón Bordone, G. (2021). *LUZ: un sistema de búsqueda sobre los archivos de la última dictadura militar* [Tesis de grado].

Chavat Pérez, F. (2022). *Modelos seq2seq para la transcripción de documentos del archivo berrutti* [Tesis de grado]. (Tesis de grado. Universidad de la República (Uruguay). Facultad de Ingeniería)

Dao, T.-A., Teranishi, H., Matsumoto, Y., y Aizawa, A. (2025). Entity-based synthetic data generation for named entity recognition in low-resource domains. En Y. I. Nakano y T. Suzumura (Eds.), *Jsai-isai* (Vol. 15692, p. 210-225). Springer. Descargado de https://books.google.com.uy/books?hl=en&lr=&id=P3VfEQAAQBAJ&oi=fnd&pg=PA210&dq=ner+synthetic+data&ots=-prHahcjBW&sig=VTVcINlA8rr2WqW7V4lmIPt-5hY&redir_esc=y#v=onepage&q&f=false

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding.* Descargado de https://arxiv.org/abs/1810.04805

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. Descargado de https://arxiv.org/abs/2010.11929

Etcheverry, L., Agorio, L., Bacigalupe, V., Barreiro, S., Bing, E., Blixen, S., ... Randall, G. (2021, February). A computational framework for the analysis of the uruguayan dictatorship archives. En *Proceedings of qurator 2021 – conference on digital curation technologies* (pp. 1–15). Berlin, Germany: Qurator 2021. Descargado de https://hdl.handle.net/20.500.12008/26651

Etcheverry, L., Rosá, A., Gómez, E., y Randall, G. (s.f.). *Ia para el procesamiento de archivos documentales y su aplicación al caso de los archivos del pasado reciente.* Descargado de https://www.fing.edu.uy/inco/grupos/gema/research-projects

Facultad de Información y Comunicación y Facultad de Ingeniería, Universidad de la República. (2025). *Cruzar – archivos del pasado reciente.* https://cruzar.edu.uy/. (Consultado: 25 de agosto de 2025)

Gamma, E., Helm, R., Johnson, R., y Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software.* Reading, MA: Addison-Wesley.

Goyal, M., y Mahmoud, Q. H. (2024). A systematic review of synthetic data generation techniques using generative ai. *Electronics*, *13*(17). Descargado de https://doi.org/10.3390/electronics13173509 doi: 10.3390/electronics13173509

He, P., Liu, X., Gao, J., y Chen, W. (2021). {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. En *International conference on learning representations.* Descargado de https://openreview.net/forum?id=XPZIaotutsD

Huang, Y., Gao, Y., y Ren, C. (2025). A survey of data augmentation in named entity recognition. *Neurocomputing*, *651*, 130856. Descargado de https://www.sciencedirect.com/science/article/pii/S0925231225015280 doi: https://doi.org/10.1016/j.neucom.2025.130856

IBM. (1987). *IBM Hardware List to 1987* (Inf. Téc.). International Business Machines Corporation.

Islam, N., Islam, Z., y Noor, N. (2017). *A survey on optical character recognition system.* Descargado de https://arxiv.org/abs/1710.05703

Jurafsky, D., y Martin, J. H. (2025). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3rd ed.). Descargado de https://web.stanford.edu/~jurafsky/slp3/ (Online manuscript released January 12, 2025)

Keraghel, I., Morbieu, S., y Nadif, M. (2024). *Recent advances in named entity recognition: A comprehensive survey and comparative study.* Descargado de https://arxiv.org/abs/2401.10825

Kim, S., Baudru, J., Ryckbosch, W., Bersini, H., y Ginis, V. (2025). *Early evidence of how llms outperform traditional systems on ocr/htr tasks for historical records.* Descargado de https://arxiv.org/abs/2501.11623

Laguna Queirolo, R. (2025). *Teacher student curriculum learning applied to optical character recognition: An analysis based on a case study* (Tesis de maestría). Universidad de la República (Uruguay), Facultad de Ingeniería. (Tesis de maestría. Universidad de la República (Uruguay). Facultad de Ingeniería)

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*(8), 707–710.

Liu, Y., He, H., Zhang, Y., Zhang, Y., Qiang, W., Hou, L., y Li, J. (2024). Monkeyocr: A precise, efficient and lightweight document parsing model.

arXiv preprint arXiv:2506.05218. Descargado de https://arxiv.org/abs/2506.05218

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2020). *Ro{bert}a: A robustly optimized {bert} pretraining approach.* Descargado de https://openreview.net/forum?id=SyxS0T4tvS

Lucas, A., Baladón, A., Pardiñas, V., Agüero-Torales, M., Góngora, S., y Chiruzzo, L. (2024, junio). Grammar-based data augmentation for low-resource languages: The case of Guarani-Spanish neural machine translation. En K. Duh, H. Gomez, y S. Bethard (Eds.), *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)* (pp. 6385–6397). Mexico City, Mexico: Association for Computational Linguistics. Descargado de https://aclanthology.org/2024.naacl-long.354/ doi: 10.18653/v1/2024.naacl-long.354

Matthews, D., y collaborators. (2024, August). Noise-induced token length increases in subword tokenization. *arXiv preprint arXiv:2408.04162*. Descargado de https://arxiv.org/pdf/2408.04162

Meta AI. (2024, julio). *Introducing Meta Llama 3.1.* Descargado 2025-09-14, de https://ai.meta.com/blog/meta-llama-3-1/ (Blog post)

Mori, S., Suen, C., y Yamamoto, K. (1992). Historical review of ocr research and development. *Proceedings of the IEEE*, *80*(7), 1029-1058. doi: 10.1109/5.156468

Munnangi, M. (2024). *A brief history of named entity recognition.* Descargado de https://arxiv.org/abs/2411.05057

Neudecker, C., Baierer, K., Gerber, M., Clausner, C., Antonacopoulos, A., y Pletschacher, S. (2021). A survey of ocr evaluation tools and metrics. En *Proceedings of the 6th international workshop on historical document imaging and processing* (p. 13–18). New York, NY, USA: Association for Computing Machinery. Descargado de https://www.primaresearch.org/www/assets/papers/HIP21_CNeudecker_OcrEvalSurvey.pdf doi: 10.1145/3476887.3476888

Nogueira, M., Etcheverry, L., y Randall, G. (2024a). *Building tools to analyze the files of the uruguayan dictatorship: Information extraction from the personal records of organización coordinadora de operaciones antisubversivas (ocoa)* (Inf. Téc.). Instituto de Computación & Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República. Descargado de https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/46460/1/NER24.pdf

Nogueira, M., Etcheverry, L., y Randall, G. (2024b). *Building tools to analyze the files of the uruguayan dictatorship: Information extraction from the personal records of organización coordinadora de operaciones antisubversivas (ocoa).* Descargado de https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/46460 (Accedido: 2025-07-10)

Papineni, K., Roukos, S., Ward, T., y Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. En *Proceedings of the 40th annual meeting on association for computational linguistics* (p. 311–318).

USA: Association for Computational Linguistics. Descargado de https://doi.org/10.3115/1073083.1073135 doi: 10.3115/1073083.1073135

Popović, M. (2015, septiembre). chrF: character n-gram F-score for automatic MT evaluation. En O. Bojar y cols. (Eds.), *Proceedings of the tenth workshop on statistical machine translation* (pp. 392–395). Lisbon, Portugal: Association for Computational Linguistics. Descargado de https://aclanthology.org/W15-3049/ doi: 10.18653/v1/W15-3049

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., y Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. En *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 101–108). Descargado de https://stanfordnlp.github.io/stanza/

Qwen Team. (2025, may). Qwen3 technical report. *arXiv preprint arXiv:2505.09388*. Descargado de https://arxiv.org/abs/2505.09388

Radford, A., Narasimhan, K., Salimans, T., y Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI*. Descargado de https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (Accessed: 2025-08-10)

Rice, S. V. (1996). *Measuring the accuracy of page-reading systems* (Ph.D. Dissertation, University of Nevada, Las Vegas). Descargado de http://dx.doi.org/10.25669/hfa8-0cqv (UNLV Retrospective Theses & Dissertations. 3014)

Sastre, I., Etcheverry, L., Rey, G., Moncecchi, G., y Rosá, A. (2025, July 15). Post-ocr correction using large language models with constrained decoding. *Research Square*. Descargado de https://doi.org/10.21203/rs.3.rs-6823036/v1 (Preprint (version 1)) doi: 10.21203/rs.3.rs-6823036/v1

Schantz, H. F. (1982). *The history of ocr, optical character recognition.* Manchester Center, VT: Recognition Technologies Users Association.

Sekine, S., y Nobata, C. (2004, mayo). Definition, dictionaries and tagger for extended named entity hierarchy. En M. T. Lino, M. F. Xavier, F. Ferreira, R. Costa, y R. Silva (Eds.), *Proceedings of the fourth international conference on language resources and evaluation (LREC'04)*. Lisbon, Portugal: European Language Resources Association (ELRA). Descargado de https://aclanthology.org/L04-1051/

Shugart, Alan. (1971). *Alan shugart of ibm introduces the 23fd floppy disk.* HistoryOfInformation.com, entry 885. Descargado de https://www.historyofinformation.com/detail.php?entryid=885

Sitios de Memoria Uruguay. (2025). *Archivo Berrutti.* https://sitiosdememoria.uy/origen/archivo-berrutti. (Consultado: 25 de agosto de 2025)

Smith, R. W. (2013). History of the tesseract ocr engine: what worked and what didn't. En *Document recognition and retrieval xx* (Vol. 8658, p. 865802).

Stabile, J., Fernández, E., y Fioritto, F. (2020a). *Informe de memoria de proyectos de procesamiento de lenguaje natural* (Inf. Téc.). Instituto de Computación, Facultad de Ingeniería, Universidad de la Re-

pública. Descargado de https://www.fing.edu.uy/inco/grupos/pln/prygrado/InformeMemoriaPLN2020.pdf

Stabile, J., Fernández, E., y Fioritto, F. (2020b). *Procesamiento de lenguaje natural (pln) para la reconstrucción de textos a partir de imágenes correspondientes a archivos históricos de la década del '70.* Descargado de https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/26094 (Accedido: 2025-07-10)

Van Strien, D., Beelen, K., Ardanuy, M., Hosseini, K., McGillivray, B., y Colavizza, G. (2020). Assessing the impact of ocr quality on downstream nlp tasks. Descargado de https://www.repository.cam.ac.uk/handle/1810/304987 doi: 10.17863/CAM.52068

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2023). *Attention is all you need.* Descargado de https://arxiv.org/abs/1706.03762

Watanabe, S. (2023). Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*. Descargado de https://arxiv.org/abs/2304.11127

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., ... Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. En *Advances in neural information processing systems (neurips).* (Preprint available at https://arxiv.org/pdf/2201.11903, accessed on 2025-08-28)

Anexo A

Acrónimos

- CER: Tasa de Error de Caracteres.
- CRF: Campo Aleatorio Condicional.
- FN: Falso Negativo.
- FP: Falso Positivo.
- GPT: Transformador Generativo Preentrenado.
- GRU: Unidad Recurrente con Compuertas.
- LLM: Gran Modelo de Lenguaje.
- LSTM: Memoria Larga a Corto Plazo.
- NER: Reconocimiento de Entidades Nombradas.
- OCR: Reconocimiento Óptico de Caracteres.
- RNN: Red Neuronal Recurrente.
- TN: Verdadero Negativo.
- TP: Verdadero Positivo.
- VIT: Visual Transformer.
- WER: Tasa de Error de Palabras (Word Error Rate).
- INCO: Instituto de Computación.
- FING: Facultad de Ingeniería de la Universidad de la República.
- GEMA: Gestión de Datos y Modelados.
- PLN: Procesamiento del Lenguaje Natural.
- IE: Extracción de Información.

- CALEN: Centro de ALtos Estudios Nacionales.
- RDF: Marco de Descripción de Recursos.
- OWL: Lenguaje de Ontología Web.
- OCOA: Órgano Coordinador de Operaciones Antisubversivas.
- LUISA: Leyendo Unidos para Interpretar loS Archivos.
- IA: Inteligencia Artificial.
- TSCL: Aprendizaje Curricular Maestro Estudiante.
- SAT: Entrenamiento Secuencial con Aumento de Datos.
- IAT: Entrenamiento Integrado con Aumento de Datos.
- LOC: Lugar.
- PER: Persona.
- ORG: Organización.
- GPU: Unidad de Procesamiento Gráfico.
- DATE: Fecha.
- BIO: Comienzo Dentro y Fuera.
- O: Sin entidad.
- LLAMA: Gran Modelo de Lenguaje de Meta AI.
- BLEU: Evaluación Bilingüe Suplente.
- CHRF: Métrica F a nivel de carácter.
- SRR: Estructura-Reconocimiento-Relación.
- PNG: Gráficos de Red Portátiles.
- JPG: Grupo Conjunto de Expertos en Fotografía.
- JSON: Notación de Objetos de JavaScript.
- COT: Cadena de Pensamiento.
- PCA: Análisis de Componentes Principales.
- CONLL: Conferencia sobre aprendizaje computacional del Lenguaje Natural.
- BERT: Representación de Codificador Bidireccional de Transformadores.
- BART: Transformadores Autorregresivos Bidireccionales.
- ROBERTA: Enfoque de Preentrenamiento de BERT optimizacion de Manera Robusta.

- DEBERTA: BERT Mejorado con Decodificación y Atención Desacoplata.
- CLS: Clasificación.
- TPE: Estimador de Parzen con Estructura de Árbol.
- IBM: Máquinas de Negocios Internacionales.
- KNN: K Vecinos Más Cercanos.
- MPL: Perceptrón Multicapapa.
- CNN: Red Neuronal Convolucional.
- HP: Hewlett-Packard.
- VAE: Autocodificador Variacional.
- GAN Red Generativa Adversativa.

Anexo B

Prompts

B.1. *Prompt* de generación de datos sintéticos

```
Tu tarea es generar [n] ejemplos que incluya relaciones entre las
↪  siguientes entidades en un texto militar uruguayo entre 1968 y 2004:
Entidades PER: [1 ejemplo de nombres de personas]
Entidades ORG: [1 ejemplo de nombres de organizaciones]

**Instrucciones:**
1. Genera un texto coherente que relacione las entidades mencionadas,
↪  utilizando frases comunes en contextos militares como "detenida en",
↪  "integrante de", "a disposición", "es detenida", "domiciliada en",
↪  etc.
2. Asegúrate de que las fechas de los sucesos estén entre 1968 y 2004 si
↪  las incluyes.
3. Divide el texto en palabras y etiqueta cada una:
    - Usa las etiquetas:
      - PER para nombres de personas.
      - ORG para organizaciones.
      - LOC para localizaciones.
      - O para palabras fuera de las entidades.
4. **Formato requerido:**
    - Cada palabra seguida de un espacio y su etiqueta (e.g., "Carlos
    ↪  PER").
    - Una línea en blanco separa los ejemplos.
5. No omitas palabras ni líneas y verifica que todas las entidades estén
↪  etiquetadas correctamente.

**Ejemplo detallado de razonamiento y salida:**

**Entrada:**
Entidades PER: Maria Perez
```

Entidades ORG: OCOA

Paso 1: Generación del texto base
"La activista Maria Perez fue detenida por el OCOA en 1973."

Paso 2: Identificación de entidades
- Persona: Maria Perez -> PER, PER.
- Organización: OCOA -> ORG.

Paso 3: Etiquetado BIO
La O
activista O
Maria PER
Perez PER
fue O
detenida O
por O
el O
OCOA ORG
en O
1973 O
. O

Salida esperada:
La O
activista O
Maria PER
Perez PER
fue O
detenida O
por O
el O
OCOA B-ORG
en O
1973 O
. O

Tu tarea:
Genera [n] ejemplos siguiendo los pasos detallados.

B.2. *Prompt* de OCR

```
Eres un asistente experto en procesamiento de imágenes y reconocimiento
↪  óptico de caracteres (OCR).
Tu tarea es transcribir textos a partir de imágenes y resultados de OCR,
↪  entregando una versión en texto plano sin errores.
No debes completar las palabras incompletas; si encuentras una palabra
↪  cortada, como 'iden-', debes mantenerla tal cual, sin intentar
↪  completarla.

Eres un asistente experto en procesamiento de imágenes y reconocimiento
↪  óptico de caracteres (OCR). \n
Tu tarea es transcribir con la mayor fidelidad posible todo el texto
↪  presente en la imagen, entregando una versión en texto plano exacta y
↪  sin errores.
Transcribe todo el texto de principio a fin, incluyendo símbolos,
↪  puntuación y mayúsculas/minúsculas tal como aparecen.
No completes ni corrijas palabras; si encuentras términos incompletos
↪  (por ejemplo: "iden-"), debes mantenerlos exactamente como se
↪  muestran, sin intentar adivinar la continuación.
No omitas texto ni agregues ningún texto adicional.
No utilices ningún tipo de formato, etiquetas o marcas (por ejemplo, no
↪  uses cursiva, negritas, subrayados ni código LaTeX); entrega
↪  únicamente texto plano.
```

Anexo C

Diccionario de transformaciones de ruido

C.1. Diccionario de ruido genérico

```
"o": ["0", "ó"],
"ó": ["6", "g"],
"0": ["0"],
"l": ["1", "i"],
"i": ["1", "í"],
"e": ["3", "€"],
"a": ["@", "á"],
"s": ["5", "$"],
"g": ["9"],
"c": ["(", "¢"],
"u": ["v"],
"n": ["m"],
"m": ["rn"],
"t": ["+"],
```