ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Extracción de Información utilizando Modelos Generativos en Documentos del Pasado Reciente

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Rodrigo Gallardo

en cumplimiento parcial de los requerimientos
para la obtención del título de
Magister en Ciencias de Datos y Aprendizaje Automático.

Directoras de Tesis

Lorena Etcheverry ..... Universidad de la República
Aiala Rosá ..... Universidad de la República

Tribunal

Jocelyn Dunstan ..... Universidad Católica de Chile
Luis Chiruzzo ..... Universidad de la República
Ignacio Ramírez ..... Universidad de la República

Directora Académica

Lorena Etcheverry ..... Universidad de la República

Montevideo
sábado 16 agosto, 2025

Extracción de Información utilizando Modelos Generativos en Documentos del Pasado Reciente, Rodrigo Gallardo.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 160 páginas.
Compilada el sábado 16 agosto, 2025.
http://iie.fing.edu.uy/

Agradecimientos

Gracias al proyecto CRUZAR y a las directoras de esta tesis por el conocimiento acumulado y las herramientas brindadas que facilitaron diversos aspectos de este trabajo.

Muchas gracias a Mateo Nogueira y su trabajo de investigación sobre las fichas de la OCOA, que permitió que este trabajo partiera desde una base sólida de información procesada.

Muchas gracias a Tryolabs por apoyar este trabajo brindando recursos de hardware que resultaron cruciales para la experimentación llevada a cabo y la construcción del sistema.

Esta página ha sido intencionalmente dejada en blanco.

A mi familia y amigos, por aguantarme en esta hermosa carrera.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El proyecto CRUZAR trabaja activamente en el desarrollo de herramientas que permitan categorizar, indexar y navegar el conocimiento presente en los documentos del archivo Berrutti. Estas herramientas habilitan un análisis mucho más rico y exhaustivo de la información existente dentro de los documentos, ayudando a investigadores y familiares en el estudio de los acontecimientos ocurridos durante la época de la dictadura uruguaya.

En este marco, se busca la construcción de un grafo de conocimiento que concentre información de interés extraída del corpus. El objetivo principal de este trabajo es la construcción de un sistema de extracción de información sobre los documentos del archivo, que permita obtener entidades, relaciones y eventos mencionados en los textos, que en el futuro alimentarán este grafo de conocimiento.

Con este objetivo en mente, se realizó un relevamiento del estado del arte para la tarea de extracción de información, donde se observó que los enfoques construidos en base a modelos de lenguaje de gran escala (Large Language Models) son capaces de obtener buenos resultados para la tarea en dominios donde los datos etiquetados son escasos, lo cual incentivó el uso de esos enfoques para el problema presente en este trabajo.

Para la construcción y evaluación del sistema de extracción de información, se construyó un conjunto de datos etiquetado con entidades, relaciones y eventos sobre un subconjunto de documentos del archivo. Este proceso requirió de la definición y ejecución de una tarea de etiquetado, la cual involucró un grupo interdisciplinario de anotadores. El resultado de la tarea de etiquetado fue un conjunto de 515 documentos anotados con un acuerdo entre anotadores y autor de 62.1 en promedio para todas las tareas, donde el acuerdo entre anotadores y autor es el promedio ponderado de la métrica F1 considerando al autor como *gold standard*. Las tareas que obtuvieron peor acuerdo son extracción de relaciones (24.6) y extracción de argumentos de eventos (44.4), y las que obtuvieron mejor acuerdo son las tareas de extracción de entidades (73.5), extracción de valores (90.5) y extracción de disparadores de eventos (77.5). Luego de un proceso de curado manual, se obtuvo un conjunto de datos de buena calidad con 403 documentos, que conforman el conjunto de datos final utilizado en este trabajo.

El sistema de extracción de información construido en base a modelos generativos alcanza un promedio ponderado de la métrica F1 de 58.4, promediado sobre todas las tareas, al utilizarse con el modelo Qwen2.5-Coder para generación de código, junto con una instrucción que formula el problema como uno de generación de código, y utilizando 3 ejemplos de entradas y salidas esperadas en la instrucción. Las tareas que obtienen los peores resultados son la extracción de relaciones (27.0) y extracción de argumentos de eventos (48.2), y las que obtienen los mejores resultados son la extracción de entidades (68.8), extracción de valores (78.3) y extracción de disparadores de eventos (69.5). Si bien estos resultados demuestran que aún hay espacio para la mejora, las

salidas del sistema son útiles y comprueban que es posible obtener un sistema de este tipo sobre los documentos del archivo. También, durante la construcción del sistema se exploraron diversas estrategias y diseños, obteniendo importantes aprendizajes sobre mejores patrones de diseño para este sistema y potenciales líneas de mejora para el futuro.

Los aportes de esta tesis son los siguientes:

- Relevamiento del estado del arte en resolución de correferencias y extracción de información.
- Definición de una ontología de dominio que guía la construcción del sistema de extracción de información y que será útil para la construcción de un grafo de conocimiento en el futuro.
- Como producto del preprocesamiento de los textos, se generó un conjunto de datos con 1000 transcripciones de fichas curadas manualmente, útil para la construcción y evaluación de sistemas de *Optical Character Resolution* (OCR).
- Desarrollo de un conjunto de datos etiquetados para la tarea de extracción de información y de un marco de trabajo para la tarea de etiquetado.
- Evaluación de diferentes estrategias para el uso de modelos generativos de lenguaje natural y código para la extracción de información.
- Diseño y desarrollo de un sistema de extracción de información para los documentos del pasado reciente.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

El proyecto CRUZAR [1] tiene el objetivo de digitalizar y sistematizar un gran número de documentos físicos relacionados a la última dictadura cívico-militar en Uruguay. Estos documentos contienen abundante información no estructurada sobre entidades, relaciones entre ellas y eventos ocurridos durante el periodo de dictadura que son de particular interés para investigadores, historiadores y familiares de desaparecidos. Debido a la gran cantidad de documentos, su heterogeneidad y falta de calidad, procesarlos manualmente requeriría de cantidades prohibitivas de esfuerzo humano. Por lo tanto, contar con un sistema informático que automatice la extracción y clasificación de esta información es de vital importancia para realizar estudios abarcativos sobre el corpus completo de documentos.

El objetivo principal de este trabajo es aprovechar los recientes avances en modelos generativos de texto para explorar la construcción de un sistema de extracción de información para los archivos del pasado reciente. En primera instancia, se busca extraer un conjunto acotado de entidades, relaciones y eventos de un subconjunto de documentos con el fin de indagar en la viabilidad de la solución.

La construcción y evaluación del sistema de extracción de información requiere de un conjunto etiquetado de entidades, relaciones y eventos. Por ende, otro de los objetivos de este trabajo es construir un conjunto de datos etiquetado sobre un subconjunto de los documentos del pasado reciente.

1.1. Descripción del problema

Estudiar y analizar los documentos del pasado reciente tiene una alta relevancia para comprender el accionar de los organismos involucrados, con el objetivo de detectar patrones de comportamiento que permitan llegar a nuevos aprendizajes sobre los acontecimientos de la época [5, 6].

Estos documentos forman parte del archivo Berrutti [2], que cuenta con aproximadamente 1500 rollos de microfilm con escaneos en blanco y negro de los documentos físicos. Entre estos rollos se suma un total de aproximadamente tres millones de imágenes de documentos de diversos tipos: recortes de prensa, fichas, cartas, transcripciones de interrogatorios y más. Estos documentos, a su vez, pueden contener texto escrito a mano o con máquina de escribir. La calidad de los escaneos en la mayoría de los casos

[1] `https://cruzar.edu.uy/`
[2] `https://sitiosdememoria.uy/origen/archivo-berrutti`

es precaria, y en consecuencia, muchas de las palabras o letras se encuentran poco visibles o ilegibles. La labor de analizar los documentos, llevada a cabo por investigadores o familiares de desaparecidos, resulta difícil ante el gran volumen de datos, su heterogeneidad, falta de calidad y la diferencia en el contexto en el que estos documentos fueron creados.

Como parte del proyecto CRUZAR, se trabaja en el desarrollo e implementación de un sistema de extracción de información textual basado en técnicas de Procesamiento de Lenguaje Natural (PLN), orientado a identificar y relacionar entidades, eventos y vínculos relevantes presentes en documentos del pasado reciente. Los resultados de estos procesos alimentarán la construcción de grafos de conocimiento navegables, que permitirán explorar las conexiones entre actores, instituciones, fechas y lugares mencionados en el corpus documental, habilitando nuevas formas de análisis e interacción con los archivos.

Si bien la construcción integral del grafo de conocimiento no forma parte del alcance de esta tesis, el presente trabajo se inscribe en este marco más amplio y realiza un aporte clave: la definición de una ontología de dominio que guía el proceso de extracción. Esta ontología provee una estructura semántica explícita que orienta la identificación de las entidades, relaciones y eventos de interés en el corpus. A futuro, se prevé utilizar esta misma ontología como esquema conceptual para organizar el grafo de conocimiento, de modo que las entidades, relaciones y eventos extraídos puedan representarse como instancias de clases y propiedades ontológicas, vinculadas explícitamente con los documentos de los cuales provienen.

Esta articulación permitirá no solo una mejor estructuración del conocimiento extraído, sino también una navegación conceptual de las colecciones documentales, por ejemplo, consultando todos los documentos que mencionan a una persona, una organización o un tipo específico de evento, promoviendo así un acceso más significativo, contextualizado y semánticamente enriquecido a las colecciones.

Previo a poder aplicar métodos de PLN sobre los documentos que permitan extraer y categorizar información, se requiere transcribir las imágenes de los documentos a texto plano, capaz de ser procesable por un computador. La labor de transcribir los documentos lleva años en desarrollo y mejora en el marco del proyecto Luisa [3], un subproyecto del proyecto CRUZAR. Sin embargo, la falta de calidad en las imágenes de los documentos hace que los algoritmos de transcripción utilizados produzcan textos también de baja calidad. Por lo tanto, es interesante investigar métodos de procesamiento de imágenes y de texto que permitan extraer información a pesar del ruido en el texto a procesar.

Los avances recientes en el área de PLN - en particular, el desarrollo y la proliferación de los modelos de lenguaje de gran escala (LLM, por su nombre en inglés, *Large Language Models*) - propiciaron el desarrollo de numerosos algoritmos de extracción de información (entidades, relaciones y eventos) a partir de texto no estructurado [4, 21, 35–37, 44, 60, 72, 75, 80, 82, 83, 88, 92]. Si bien otros tipos de algoritmos para resolver estas tareas llevan décadas en desarrollo, nuevos algoritmos construidos en base a LLMs logran obtener buenos resultados en problemas donde los datos etiquetados son escasos, como es el caso de los documentos del pasado reciente [82, 88]. Mientras que antes se necesitaban centenares o miles de documentos etiquetados para entrenar algoritmos de Aprendizaje Automático clásico o Deep Learning, hoy en día solo se requieren algunas decenas de ejemplos para obtener resultados útiles, lo cual reduce enormemente el esfuerzo humano involucrado en exploración y desarrollo.

El objetivo principal de este trabajo es aprovechar los recientes avances en modelos generativos de texto para explorar la construcción de un sistema de extracción de información para los archivos del pasado reciente. En primera instancia, se busca

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

extraer un conjunto acotado de entidades, relaciones y eventos de un subconjunto de documentos con el fin de indagar en la viabilidad de la solución. Este subconjunto de datos son las fichas gestionadas por el Organismo Coordinador de Operaciones Antisubversivas (OCOA).

En los documentos, pueden ocurrir diversas menciones a la misma entidad. Para las personas, es normal que aparezcan mencionadas con sus apodos, pronombres, títulos u otras formas. Resolver las correferencias entre menciones es importante para lograr asociar entidades a relaciones y eventos correctamente. Por lo tanto, otro objetivo de este trabajo es analizar la necesidad de incluir una etapa previa de procesamiento que intente resolver las correferencias en el texto.

El primer paso para cumplir estos objetivos consistió en realizar un relevamiento de las tecnologías y métodos actuales para la extracción de información en textos. En una primera iteración, se hizo un estudio de las tecnologías existentes para la resolución de correferencias, con el objetivo de resolver el problema en dos pasos: primero extraer todas las menciones a entidades y los enlaces de correferencia entre las mismas, y luego usar esta información para extraer relaciones y eventos. Luego de un estudio detallado de la literatura se observó que las soluciones para el problema de resolución de correferencia usualmente requieren de una enorme cantidad de datos etiquetados y recursos para entrenar modelos de aprendizaje automático tradicionales, por lo cual la primera idea de solución posiblemente era inviable. En esta investigación de la literatura se observó que un gran número de trabajos recientes utilizan modelos generativos para la extracción de información. Estos modelos, pre-entrenados sobre enormes volúmenes de textos, son capaces de modelar patrones típicos dentro del lenguaje y de resolver correferencias de forma implícita. Esto hace que sean particularmente útiles para extraer información en textos, incluso resolviendo enlaces de correferencia en el proceso. Esta ventaja, sumada al hecho de que se ha demostrado que se requieren de pocos datos para tener una solución funcional, hicieron que se optara por construir un sistema de extracción de información utilizando modelos generativos. El relevamiento del estado del arte sobre estas tecnologías y tareas se encuentra en el capítulo 2.

La construcción y evaluación del sistema de extracción de información requiere de un conjunto etiquetado de entidades, relaciones y eventos. A su vez, como se ha visto en trabajos recientes, la calidad de los datos utilizados para construir sistemas con modelos generativos es de suma importancia para obtener buenos resultados con un volumen pequeño de datos [73]. Por ende, otro de los objetivos de este trabajo es construir un conjunto de datos etiquetado sobre un subconjunto de los documentos del pasado reciente. Este conjunto de datos fue etiquetado con un equipo interdisciplinario de estudiantes y voluntarios en un esfuerzo coordinado. Si bien trabajos previos del proyecto CRUZAR han etiquetado datos de entidades [10], este es el primer conjunto de datos construido para las tres tareas unificadas, y plantea un marco de trabajo y etiquetado de datos que se puede reutilizar para expandir la ontología de los datos extraídos o el corpus de texto en futuras iteraciones. Una descripción del conjunto de datos y de la tarea de etiquetado se encuentra en el capítulo 3.

Una cualidad necesaria de la solución a desarrollar es que deba ser posible expandirla para procesar una mayor cantidad de documentos - posiblemente de distintas características - y para extraer un conjunto distinto de datos. Con esto en mente, se diseñó una solución con una arquitectura de pipeline modular, lo cual facilita la exploración de distintas configuraciones y abre las puertas a futuros trabajos que quieran aplicar esta solución a nuevos dominios. Una descripción de la arquitectura de la solución y los componentes desarrollados se encuentra en el capítulo 4.

En el capítulo 5 se evalúan las distintas variantes de la solución y se analizan los resultados obtenidos en profundidad con el fin de hallar una configuración que resuelva satisfactoriamente el problema. En el capítulo 6 se plantean las conclusiones obtenidas

del trabajo y se proponen posibles líneas de trabajo a futuro.

Las contribuciones de esta tesis son las siguientes:

- Relevamiento del estado del arte en resolución de correferencias y extracción de información.
- Definición de una ontología de dominio que guía la construcción del sistema de extracción de información y que será útil para la construcción de un grafo de conocimiento en el futuro.
- Como producto del preprocesamiento de los textos, se generó un conjunto de datos con 1000 transcripciones de fichas curadas manualmente, útil para la construcción y evaluación de sistemas de *Optical Character Resolution* (OCR).
- Desarrollo de un conjunto de datos etiquetados para la tarea de extracción de información y de un marco de trabajo para la tarea de etiquetado.
- Evaluación de diferentes estrategias para el uso de modelos generativos de lenguaje natural y código para la extracción de información.
- Diseño y desarrollo de un sistema de extracción de información para los documentos del pasado reciente.[4]

[4] El código de la solución y los experimentos realizados se encuentra disponible en `https://gitlab.fing.edu.uy/mh/information_extraction`.

Capítulo 2

Marco Teórico

La tarea de extraer y clasificar información a partir de un texto lleva décadas en desarrollo. Los avances en las áreas de Aprendizaje Automático y Aprendizaje Profundo permitieron la creación de múltiples algoritmos que permiten extraer distintos tipos de información de un texto, requiriendo altos volúmenes de datos para entrenar estos algoritmos. Recientemente, nuevos métodos que utilizan modelos de lenguaje de gran escala (LLM, por su nombre en inglés, *Large Language Model*) permiten construir soluciones de extracción de información con escasos o nulos datos de entrenamiento, lo cual facilita la aplicación de estos métodos en casos donde no se cuenta con datos etiquetados.

En este capítulo, se introducen los conceptos más relevantes utilizados en el resto del trabajo y se detallan los algoritmos para la extracción de información desarrollados en los últimos años.

2.1. Conceptos Preliminares

En esta sección se introducen algunos conceptos importantes del área PLN que facilitan la lectura y comprensión del resto del trabajo.

2.1.1. Tokenizadores y modelos de lenguaje

Previo a poder ser procesado por un modelo de lenguaje, un texto debe ser descompuesto en unidades mínimas llamadas *tokens*. Estas unidades pueden ser palabras, subpalabras o caracteres, dependiendo de la estrategia de tokenización que se utilice. Luego, se identifican todos los distintos tokens existentes en un corpus y se les asigna a cada uno un identificador numérico, creando lo que se llama un *vocabulario*. Esto permite traducir un texto en una cadena de tokens y, utilizando el vocabulario construido, en una cadena de identificadores numéricos, que son utilizados por el modelo de lenguaje para procesar el texto. El proceso de segmentar un texto en tokens se lleva a cabo por un componente vital de los sistemas de PLN actuales llamado Tokenizador [27].

Los modelos de lenguaje definen una distribución de probabilidad sobre todos los tokens de un vocabulario, condicionado sobre una secuencia de tokens ya vistos, en un idioma dado. Esto permite que se puedan utilizar para encontrar el siguiente token más probable en una secuencia de tokens, o para asignar probabilidades de ocurrencia a secuencias completas [27]. Históricamente, los primeros modelos de lenguaje se basaban en enfoques estadísticos (por ejemplo, los modelos *n-gram*), pero en la última

década han sido desplazados por modelos de redes neuronales profundas entrenados con grandes corpus de texto, los cuales obtienen un mejor desempeño que los enfoques estadísticos en diversas tareas de PLN [93].

2.1.2. Modelos de lenguaje de gran escala (LLM)

Se denomina modelos de lenguaje de gran escala (LLM, por su nombre en inglés *Large Language Model*) a modelos de lenguaje que cuentan con un número de parámetros en el orden de los millones o miles de millones. Estos modelos están en su mayoría creados utilizando la arquitectura Transformer [68]. Esta arquitectura introdujo una red neuronal basada en mecanismos de *self-attention*, eliminando por completo la necesidad de recurrencias o convoluciones en el modelado de secuencias. Esta arquitectura se consolidó rápidamente como la columna vertebral de los modelos de lenguaje modernos, reemplazando a las redes neuronales recurrentes (RNN, por su nombre en inglés *Recurrent Neural Network*) y a las redes neuronales convolucionales (CNN, por su nombre en inglés *Convolutional Neural Network*) en múltiples aplicaciones.

El uso más popular de estos modelos es la generación de texto, el cual consiste en proveer al modelo una instrucción (o *prompt*, por su nombre en inglés) indicando una tarea a realizar e información relevante para cumplirla o proveer un texto a completar, y utilizar la capacidad del LLM de predecir el siguiente token en la secuencia para iterativamente generar una respuesta de texto completa a la instrucción dada. Este proceso es llamado *autoregressive generation* [27]. Existen diversas estrategias y diversos parámetros (por ejemplo, el parámetro "temperatura") que se pueden modificar para seleccionar el siguiente token utilizando la distribución de probabilidad entrenada por el modelo y generar la secuencia de tokens que conforman la respuesta. Modificaciones en estas estrategias o parámetros pueden ayudar a obtener respuestas que simulen una respuesta humana o que sean más consistentes y repetitivas, dependiendo de la necesidad de cada caso de uso.

La construcción de la instrucción a proveer al modelo no es trivial, y se ha demostrado que distintas variantes de instrucción para la misma tarea pueden obtener resultados diferentes, a veces determinando si el modelo puede efectivamente cumplir la tarea [59]. El formato de la instrucción, la claridad en la escritura, las directivas utilizadas, el vocabulario elegido, el uso de ejemplos y la extensión de la instrucción son factores que pueden alterar el resultado producido. Una de las metodologías utilizadas para encontrar la instrucción que mejor se ajuste para una tarea en particular es llamada *prompt engineering*, la cual consiste en un proceso altamente experimental, en el cual se prueban distintas variantes de instrucción, se evalúan los resultados obtenidos y se itera el proceso durante un tiempo acotado. Métodos automáticos, también llamados *prompt optimizers*, han cobrado relevancia en el presente [24, 58], aunque estos enfoques siguen en desarrollo.

Los avances científicos han demostrado que al aumentar considerablemente el tamaño de los LLM o entrenarlos con grandes volúmenes de datos, estos modelos no solo mejoran significativamente su rendimiento, sino que también manifiestan habilidades emergentes [93]. Por definición, las habilidades emergentes son características de los LLM que no están presentes en modelos más pequeños y que solo se obtienen con una mayor escala [78]. Una de estas habilidades emergentes es la capacidad de generalizar sobre tareas o datos no vistos en el conjunto de entrenamiento. Esto permite que los LLM, con instrucciones adecuadas y sin la necesidad de un entrenamiento dedicado, obtengan resultados accionables en tareas de nicho o sobre dominios específicos donde los datos escasean [93].

Si bien los LLM han demostrado la capacidad de reproducir conocimiento aprendido durante el pre-entrenamiento y obtenido de los corpus utilizados para entrenar-

los, una de sus limitantes es la incapacidad de acceder a información actualizada, no presente en los datos de entrenamiento, y de seleccionar esta información de forma inteligente para cumplir tareas aguas abajo, lo cual es una fuerte limitante para tareas que requieren de un alto nivel de conocimiento de cierto dominio. Para solventar estas limitantes se desarrolló la técnica *Retrieval Augmented Generation* (RAG) [34]. Esta técnica, que ha sido ampliamente adoptada, implica almacenar conocimiento de dominio en una base de datos y, ante una consulta o solicitud de tarea al LLM, recuperar información relevante de la base de datos e incluirla en la instrucción. Esta estrategia no requiere de adaptar el modelo al nuevo dominio, por lo cual es fácil de aplicar y se puede realizar incluso con escasos datos. Se ha demostrado que esta técnica permite que los LLM generen respuestas más específicas, correctas y basadas en documentos que las sustentan.

2.1.3. Tipos de entrenamiento y uso de ejemplos

Existen diferentes paradigmas para entrenar o adaptar modelos de lenguaje a tareas específicas. El enfoque tradicional es el *fine-tuning* supervisado, que consiste en ajustar un subconjunto de los pesos de un modelo pre-entrenado usando un conjunto de datos etiquetados de la tarea objetivo [27]. Este proceso permite aprovechar el conocimiento general adquirido durante el pre-entrenamiento y lo especializa para una tarea o dominio de conocimiento no vistos durante el entrenamiento.

Por otro lado, los enfoques *zero-shot* y *few-shot* se refieren a la capacidad de aplicar modelos a nuevas tareas o dominios con pocos o nulos datos de entrenamiento específicos para la tarea, aprovechando las habilidades emergentes mencionadas previamente [8].

En el caso *zero-shot*, el modelo recibe únicamente una instrucción describiendo la tarea en lenguaje natural, sin ejemplos concretos, y aún así es capaz de realizarla apoyándose en el conocimiento general que posee.

En el caso *few-shot*, se cuenta con un conjunto acotado de datos etiquetados, con un tamaño en el orden de las decenas, y se utilizan algunos pocos ejemplos para optimizar el modelo para una tarea en particular. Un ejemplo de estrategia few-shot es la estrategia In-Context Learning (ICL) [17], en la cual se le proporciona al modelo uno o muy pocos ejemplos de la tarea como parte de la instrucción, de modo que pueda inferir el patrón deseado a partir de un subconjunto mínimo, que no suele necesitar más de 20 ejemplos. Estos ejemplos son pares de entrada y salida esperada, y se espera que el modelo pueda reconocer los patrones con los cuales debe generar la salida para la entrada provista en la instrucción.

Los LLMs modernos pueden desempeñar tareas complejas bajo estos esquemas sin necesitar un reentrenamiento completo o fine-tuning. Este uso reduce significativamente la dependencia a grandes conjuntos de datos etiquetados, lo cual simplifica la construcción de sistemas basados en LLMs para tareas o dominios donde los datos son escasos. Sin embargo, si se necesita incrementar la efectividad de la solución sobre una tarea específica, es muy probable que haya que recurrir a técnicas de fine-tuning o re-entrenamiento.

2.1.4. Modelos de embeddings y bases de datos vectoriales

Los modelos de embeddings se refieren a técnicas que convierten unidades lingüísticas (palabras, oraciones o documentos) en vectores numéricos de dimensión fija [27]. Las representaciones vectoriales forman parte de un espacio continuo de alta dimensión, diseñado de modo que elementos con un significado semántico similar se encuentren cercanos entre sí en dicho espacio. Esto permite construir sistemas de búsqueda

semántica por similitud: ante un texto de entrada, se calcula su representación vectorial y se buscan vectores cercanos en el espacio representado.

En la actualidad, se han desarrollado múltiples modelos de embeddings basados en la arquitectura Transformer presentada previamente. Estos algoritmos permiten la construcción de embeddings contextuales, los cuales permiten capturar el significado semántico de cada token según la oración completa en la que participa, de manera que palabras o frases ambiguas (por ejemplo, "banco") poseen representaciones distintas dependiendo de dónde se utilicen. Uno de los modelos más potentes desarrollados siguiendo esta arquitectura es BGE-M3 [12], el cual combina distintas técnicas de pre-entrenamiento y distilación para construir un modelo de embeddings que opera sobre más de 100 idiomas, funciona sobre textos de distintos largos e incluso textos extensos de hasta aproximadamente 8000 tokens, y está optimizado para utilizarse en sistemas de búsqueda por similitud de vectores.

Las bases de datos vectoriales son sistemas de almacenamiento y búsqueda optimizados para manejar grandes colecciones de vectores numéricos, usualmente producto de convertir elementos textuales o multimedia en vectores utilizando modelos de embeddings. Estos sistemas indexan y almacenan datos no estructurados junto con sus representaciones vectoriales y permiten acceder a ellos eficientemente mediante búsquedas de similitud.

Estas bases de datos han resultado cruciales para la búsqueda semántica a gran escala y para la integración de conocimiento en sistemas de PLN, en técnicas como RAG [34] o ICL [17].

2.2. Extracción de Información

La tarea de Extracción de Información (IE, por su nombre en inglés, *Information Extraction*) consiste en la obtención y clasificación de información estructurada como entidades, relaciones entre entidades y eventos a partir de texto no estructurado en lenguaje natural [27].

En la literatura IE se suele descomponer en subtareas conocidas como Extracción de Entidades Nombradas (NER, por su nombre en inglés, *Named-Entity Recognition*), Extracción de Relaciones (RE, por su nombre en inglés, *Relation Extraction*) y Extracción de Eventos (EE, por su nombre en inglés, *Event Extraction*). Cada uno de estos subgrupos se puede descomponer en tareas más específicas o variantes dependiendo del problema en cuestión.

- NER: consiste en extraer segmentos del texto que corresponden a menciones a entidades y en clasificar el tipo de dichas menciones. Por ejemplo, identificar las menciones a personas u organizaciones.
- RE: consiste en extraer tuplas que corresponden a relaciones entre menciones a entidades. Por ejemplo, puede ser de interés buscar la relación "trabaja en" que tiene como origen una persona y como destino una organización. Esta relación se suele representar como una tupla (persona, trabaja en, organización).
- EE: esta tarea se suele dividir en dos: identificar y clasificar los disparadores de eventos (la palabra o secuencia que claramente expresa la ocurrencia del evento) y los argumentos del evento (segmentos de texto que aportan más información sobre el evento). Los nombres de estas subtareas son Detección de Eventos (ETD, por su nombre en inglés, *Event Trigger Detection*) y Detección de Argumentos de Eventos (EAD, por su nombre en inglés, *Event Argument Detection*).

En la figura 2.1 se pueden visualizar las salidas de las tareas NER, RE y EE para un mismo ejemplo de entrada. Para la tarea NER, la salida son spans de texto clasificados

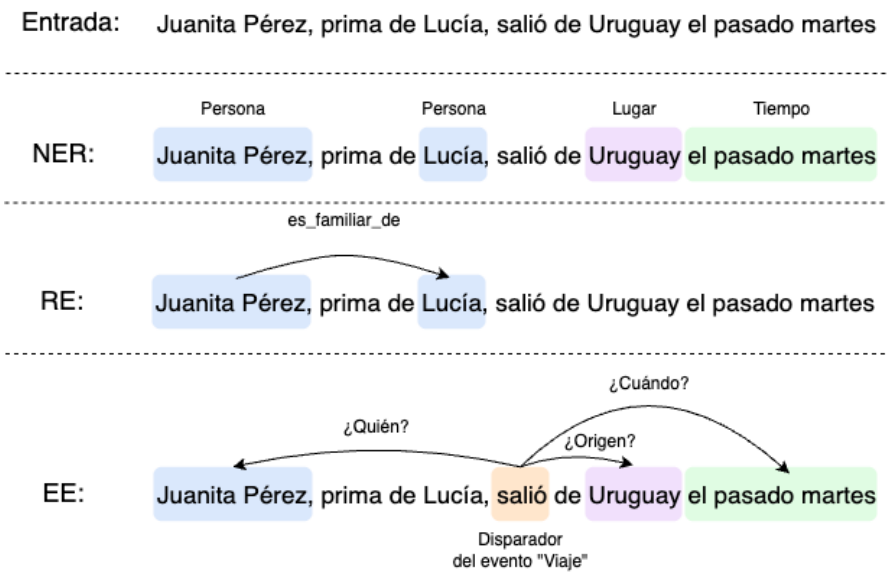


Figura 2.1: Salida de las tareas NER, RE y EE para una misma entrada de ejemplo. Para la tarea NER, se identifican las entidades Persona y Lugar y el valor Tiempo. Para la tarea RE, se identifica la relación `es_familiar_de`. Para la tarea EE, se identifica el evento "Viaje" y los argumentos asociados.

según alguno de los tipos de entidad (Persona, Lugar o el valor Tiempo). Para la tarea RE, la salida son pares de entidades clasificados según alguno de los tipos de relación; en el ejemplo, se identifica que el par de entidades "Juanita Pérez" y "Lucía" forman parte de una relación `es_familiar_de`. Para la tarea EE, primero se identifica un span de texto que actúa como disparador del evento: en el ejemplo, la palabra "salió" implica un evento de tipo Viaje. Luego, se identifican los argumentos de este evento. Los argumentos son entidades o valores que aportan información particular sobre el evento; en este caso, se identifica quién fue el sujeto principal del evento ("Juanita Pérez"), en qué lugar se originó el evento ("Uruguay") y cuándo ocurrió ("el pasado martes").

La salida de las tareas de IE suele ser información estructurada que permite realizar consultas automáticas y eficientes sobre un conjunto de textos desestructurados. Por esto, se suele hablar de IE en la etapa de construcción de un Grafo de Conocimiento (KG, por su nombre en inglés, *Knowledge Graph*). Un KG almacena información de diversas entidades y relaciones semánticas entre estas para facilitar el análisis de la información, permitiendo detectar patrones en las entidades y relaciones o resolver consultas complejas sobre el grafo. Para popular dicho grafo, primero es necesario extraer las entidades y relaciones del texto, donde las técnicas de IE entran en juego [82].

Las tareas NER, RE y EE llevan décadas en desarrollo independiente. Esto ha permitido lograr que haya buenas soluciones específicas para resolver cada una de esas tareas o algunas de sus variantes de forma independiente. Sin embargo, integrar estas soluciones en un único sistema de IE suele ser una tarea engorrosa, debido a que cada tarea tiene sus propios conjuntos de datos con sus propios formatos de anotación, o cada tarea tiene su propia estructura de entrada o salida, o cada solución utiliza distintas tecnologías o herramientas. A su vez, el desarrollo independiente de cada

tarea es subóptimo en casos donde hay escasez de datos.

Las tareas individuales pueden aprovechar el conocimiento aprendido por las otras tareas para potenciar sus propios resultados, y así compartir entre todas las tareas las habilidades intrínsecas necesarias para extraer información (extracción de texto y estructura, clasificación, relacionamiento de entidades y más). Por esto, en [41] se propone la tarea de *Universal Information Extraction* (UIE), la cual consiste en crear un marco de trabajo unificado para resolver varias tareas de IE con un mismo sistema. Los autores argumentan que utilizar las mismas operaciones de transformación y estructura para todas las tareas ayuda a generar estructuras más fácilmente y a compartir habilidades de IE entre todas las tareas. Esto es provechoso en contextos donde los datos son escasos, y a su vez facilita la integración y resolución de varias tareas de IE en simultáneo. Su modelo, pre-entrenado sobre corpus de texto generales y conjuntos de datos de IE, alcanza resultados de estado del arte, incluso mejores que modelos específicamente entrenados para cada tarea por separado. Este trabajo sienta las bases para diversos trabajos enfocados en UIE, los cuales se discuten a continuación.

2.2.1. Enfoques tradicionales

Los enfoques tradicionales (también llamados "enfoques discriminativos" en algunos trabajos de la literatura) usan técnicas de Aprendizaje Automático, Aprendizaje Profundo y algoritmos de PLN conocidos para resolver las tareas de IE. Un factor en común entre los algoritmos que siguen este enfoque es que requieren conjuntos anotados de datos de gran tamaño para entrenar componentes hechos a medida para cada solución. La falta de datos anotados para el problema a resolver en este trabajo hace que utilizar este tipo de algoritmos sea inviable. De todas formas, con el fin de presentar un marco teórico más completo de las herramientas existentes para IE, se incluye un breve resumen de algunos de los algoritmos que siguen el enfoque tradicional.

Generalmente, estos sistemas suelen trabajar a nivel de tokens o spans del texto como unidades mínimas de trabajo. Luego, suelen estimar funciones que para cada unidad o par de unidades calculan puntajes que permiten determinar si una unidad es una entidad o parte de una entidad de cierto tipo, o si un par de unidades conforman una relación. Aunque el uso de enfoques tradicionales (no generativos) es común en sistemas específicos para cada tarea de IE, el uso de enfoques tradicionales para UIE es una línea que apenas se ha explorado. A continuación se describen algunos trabajos que utilizan este enfoque para intentar resolver el problema de UIE.

[40] propone resolver la tarea mediante un método llamado *Unified Semantic Matching* (USM), en el cual se descomponen las tareas de IE en operaciones básicas de enlazado a nivel de tokens. Mediante el entrenamiento de redes profundas, se crea un modelo que detecta estas operaciones y permite reconstruir las estructuras de información a extraer. En primera instancia, se reconstruyen spans que identifican menciones a entidades. Luego, estos spans se enlazan con etiquetas de entidades o se enlazan entre sí para formar relaciones o eventos. Todo el algoritmo se basa en el enlazado entre pares de estructuras.

Los autores de [95] argumentan que el método anterior es limitado a la hora de extenderlo a tareas más específicas de extracción de información como *Discontiunous NER*, que necesitan clasificar o asociar múltiples estructuras. Por eso proponen Mirror, el cual es un framework que desarrolla la idea anterior y modela el problema como uno de extracción de un grafo compuesto por múltiples spans. A diferencia del anterior, hay distintos enlaces a nivel de span que permiten luego decodificar el grafo en las estructuras específicas de cada tarea.

RexUIE [39] es otro método alternativo para solventar las carencias de USM. En este, se extraen spans recursivamente a partir de un esquema utilizando operaciones

de enlazado a nivel de token similares a las de USM. La extracción recursiva permite utilizar los spans extraídos en el paso anterior y de esa forma extraer esquemas más complejos que en USM.

Al igual que Mirror [95], otros trabajos [38, 67, 71, 77] han intentado modelar el problema de UIE como uno de construcción, optimización y decodificación de grafos.

En [71] se propone DyGIE++. En este enfoque se extraen posibles spans del texto, se obtienen sus embeddings con un modelo de lenguaje y se construye un grafo con dichos spans. Luego, mediante operaciones de pasaje de mensajes y enlaces específicos para cada tarea, se actualizan los embeddings de cada span, así también codificando información de cada tarea y de spans relacionados. Luego, se utilizan funciones de puntaje sobre las representaciones finales de cada span para obtener la salida de cada tarea.

OneIE [38] propone un modelado en base a grafos distinto a los métodos anteriores. En su grafo, las menciones a entidades y los disparadores de eventos son nodos, y las relaciones entre entidades o entre disparadores y argumentos de eventos son enlaces. Como primer paso, se obtienen embeddings de todos los tokens y se detectan menciones a entidades y disparadores de eventos. Luego, usando funciones de puntaje, se identifican tipos de las menciones, relaciones entre menciones y relaciones entre disparadores y argumentos, para construir una versión inicial del grafo. Utilizando *features* globales construidas manualmente, optimizan este grafo para mejorar la interacción entre distintas tareas o entre distintas instancias de una misma tarea. Finalmente, utilizan un algoritmo de *beam-search* para decodificar el grafo final que produce los resultados para cada tarea.

Los creadores de FourIE [67] argumentan que DyGIE++ y OneIE no modelan la interdependencia entre tareas explícitamente y solo incluyen dicha información durante la inferencia, y no en el entrenamiento. Para solventar esto, crean un grafo que modela la interacción entre las distintas tareas, que luego procesan con una *Graph Convolutional Neural Network* para mejorar la representación de cada una de las instancias de una tarea en base a su relación con las otras. También utilizan un grafo con dependencias entre los tipos de entidades, relaciones y eventos para regularizar el entrenamiento.

2.2.2. Enfoques generativos

Los recientes avances en el área de los LLMs propiciaron la investigación de nuevos métodos. Estos modelos fueron entrenados en grandes volúmenes de datos, de contextos variados, y una multitud de tareas, lo cual permite que puedan generalizar a tareas o datos no vistos durante el entrenamiento y utilizarse en modalidad few-shot o zero-shot. Numerosos trabajos recientes utilizan estos modelos para las tareas de IE, en particular, NER, RE y EE.

Recientemente se han realizado relevamientos del uso de modelos generativos para extracción de información [82, 88], donde se proponen distintas taxonomías para clasificar los distintos sistemas planteados. En particular, este resumen se inspira en la taxonomía planteada en [82], utilizando las siguientes dimensiones para comparar sistemas:

- Utilización de datos: se compara el uso de estrategias zero-shot, few-shot (una de ella siendo la estrategia ICL) o fine-tuning supervisado.
- Tipo de LLM utilizado: se diferencia entre soluciones que utilizan LLMs entrenados y diseñados para procesar y generar lenguaje natural (NL-LLM) o código de programación (Code-LLM).

- Tipo de instrucción utilizada: se diferencia entre soluciones que utilizan instrucciones de lenguaje natural o de código para instruir a los LLMs en extraer la información.
- Diseño de instrucciones: se destacan diseños de instrucciones más avanzados que buscan mejorar el desempeño en las distintas tareas.

Utilización de datos

Algunos trabajos como [35, 80, 92] intentan explotar las habilidades emergentes demostradas por LLMs modernos y usarlos en modalidad zero-shot para resolver la tarea de extracción de información.

Otros trabajos se concentran en la modalidad few-shot, en la cual se cuenta con un volumen pequeño de datos que se puede utilizar para ajustar los sistemas y mejorar los resultados. En este caso, destaca la técnica ICL, la cual consiste en proveer ejemplos de entrada y salida esperada en la instrucción al modelo. Trabajos como [4, 21, 36, 44, 92] aplican esta técnica utilizando LLMs sin ningún tipo de pre-entrenamiento o fine-tuning y demuestran que logran mejores resultados. En [44] los autores también proveen ejemplos "negativos" de salidas incorrectas en una técnica que llaman *Contrastive In-Context Learning*, concluyendo que esta estrategia es incluso mejor que solo proveer ejemplos positivos o correctos.

La selección de los ejemplos a proveer no es trivial. En los trabajos [21,44] se seleccionan los ejemplos a incluir en la instrucción en base a similitud de *embeddings* entre la oración de entrada y un conjunto de ejemplos de entrada y salida de entrenamiento, siendo esta estrategia muy similar a la técnica RAG. Esto hace que los ejemplos seleccionados para la instrucción tengan una similitud semántica alta con la entrada, lo cual, en última instancia, mejora los resultados obtenidos por el LLM.

Trabajos como [37, 60, 72, 75, 83] realizan fine-tuning de LLMs sobre conjuntos de datos de IE existentes para mejorar la capacidad de estos modelos de resolver la tarea.

También, [37, 72] realizan una etapa de pre-entrenamiento de los LLMs sobre conjuntos de datos de código o de generación de esquemas o información estructurada. El argumento es que esta etapa de pre-entrenamiento permite dotar a los LLMs, entrenados sobre lenguaje natural desestructurado, con la capacidad de entender y generar estructuras de forma más sencilla, lo cual afecta positivamente las tareas de extracción de información.

Cuando se cuenta con suficientes ejemplos de entrenamiento, las estrategias de pre-entrenamiento y fine-tuning son útiles para obtener mejores resultados en las tareas de IE. Sin embargo, si se cuenta con un conjunto de algunas decenas de ejemplos, la estrategia ICL, combinada con búsqueda de ejemplos en base a similitud de embeddings, puede ser útil para obtener mejores resultados sin la necesidad de fine-tuning o pre-entrenamiento.

Tipo de LLM utilizado

En los trabajos del área se hace una distinción entre LLMs entrenados y diseñados para generar lenguaje natural, llamados Natural Language LLM (NL-LLM), o los entrenados y diseñados para generar código de programación, llamados Code-LLM. Naturalmente, ambos tipos de modelos fueron entrenados para tareas distintas, y esto les da a ambos características diferentes que pueden ser de utilidad para resolver la tarea de extracción de información. Por un lado, los NL-LLM están pre-entrenados sobre grandes volúmenes de lenguaje natural y entrenados para tareas comunes de comprensión y razonamiento sobre el lenguaje, lo cual los provee de cualidades útiles para comprender el texto y extraer información del mismo. Por otro lado, los Code-LLM resultan de utilizar NL-LLM como modelos base, pre-entrenarlos sobre grandes

volúmenes de código y optimizarlos para tareas de generación y compresión de código de programación [66, 86, 94]. Dada la naturaleza estructurada de los lenguajes de programación y los tipos de datos que se manejan dentro de estos, este tipo de modelos es particularmente bueno para generar salidas estructuradas [76]. También, dado que estos modelos parten de un NL-LLM y que fueron diseñados para resolver tareas de programación, son capaces de entender las instrucciones en lenguaje natural dadas por el usuario, por lo que estos modelos también tienen capacidad de comprender texto en lenguaje natural.

Algunos trabajos [4, 21, 36, 37, 44, 60, 76] utilizan Code-LLMs para IE, valiéndose de los argumentos expuestos previamente, y demostrando que en general estos modelos superan a los NL-LLMs si no se usa ninguna otra estrategia para mejorar los resultados. En [36] los autores prueban que Code-LLMs también son mejores para casos few-shot, y comprueban que el uso de Code-LLMs e instrucciones de código suele ser mejor que el uso de NL-LLMs e instrucciones escritas en lenguaje natural.

Otros trabajos [41, 72, 75, 80, 83] utilizan NL-LLMs junto con otras estrategias para mejorar los resultados. Por ejemplo, en [72, 75] se hacen entrenamientos específicos de los modelos para mejorar la capacidad de los mismos de predecir estructuras, lo cual beneficia las tareas de IE.

El uso de Code-LLMs o NL-LLMs, combinados con distintas estrategias, se encuentra dividido en los trabajos previos, sin un acuerdo popular que permita concluir que un tipo de LLM sea mejor para toda tarea de IE o para cualquier dominio. Por ende, es necesario experimentar con ambos tipos de LLM para cada problema particular.

Tipo de instrucción utilizada

Otra variante es el tipo de instrucción usada, donde se consideran dos tipos: de código o de lenguaje natural.

En la instrucción de código, se transforma el problema de extracción de información en uno de generación de código, donde se pueden utilizar estructuras de datos especificadas en la instrucción para instanciar la información a extraer [4, 21, 36, 37, 44, 60, 76]. El uso de estructuras de datos, como por ejemplo, clases de Python o modelos de Pydantic [**?**], permite definir la información a extraer en un formato legible, flexible y fácil de instanciar con otros programas.

El ejemplo 2.1 muestra la instrucción de código utilizada para la tarea NER en [36]. En este caso, la instrucción se formula como una función de Python, y las entidades a extraer se instancian como diccionarios con los atributos “text” y “type”.

Ejemplo 2.1: Ejemplo de instrucción de código utilizada en [36] para la tarea NER. Las líneas luego del comentario “# extracted named entities” se espera que sean generadas por el modelo como salida.

```
def named_entity_recognition(input_text):
    """ extract named entities from the input_text . """
    input_text = "Steve became CEO of Apple in 1998 ."
    entity_list = []
    # extracted named entities
    entity_list.append({"text": "Steve", "type": "person"})
    entity_list.append({"text": "Apple", "type": "organization"})
```

En la instrucción de lenguaje natural, se indican las clases a extraer y se solicita que la información se extraiga en formato de tuplas u oraciones que representen la información extraída [35, 36, 72, 75, 80, 83, 92]. El ejemplo 2.2 muestra la instrucción de lenguaje natural utilizado para la tarea de RE en [75].

Ejemplo 2.2: Instrucción utilizada en [75] para la tarea RE.

```
Given a sentence, please extract the subject and object
containing a certain relation in the sentence according
to the following relation types, in the format of
"relation1: word1, word2; relation2: word3, word4"

Options: used for, part of, located in

Answer:
```

En [36] los autores concluyen que el uso de instrucciones de código y Code-LLMs produce mejores resultados que el uso de NL-LLMs e instrucciones de lenguaje natural. En [76] los autores demostraron que para algunos Code-LLMs, con la cantidad suficiente de ejemplos de ICL, una instrucción de lenguaje natural puede alcanzar resultados similares a una instrucción de código, pero que en la mayoría de los casos, la instrucción de código es superior. Los casos en los cuales la instrucción de lenguaje natural obtiene mejores resultados suelen ser cuando el Code-LLM con el que es utilizada fue optimizado utilizando la estrategia Reinforcement Learning from Human Feedback (RLHF) [29], en la cual el LLM intenta optimizar una función de recompensa con un objetivo dinámico, que se alinea con los intereses y valores de usuarios humanos. Esta estrategia ha demostrado ser útil para obtener respuestas mejor recibidas por los usuarios y para alinear las salidas del LLM con ciertos estándares de conducta [3].

Igual que con los tipos de LLM, tampoco hay un acuerdo popular sobre qué tipo de instrucción es mejor para todo problema de IE o para cualquier dominio. Tampoco está claro si un tipo de instrucción opera mejor con un cierto tipo de LLM, aunque se puede intuir que una instrucción de código opere mejor con un Code-LLM que con un NL-LLM y viceversa para la instrucción de lenguaje natural. Por estas razones, también es necesario experimentar con ambos tipos de instrucción, combinados con ambos tipos de LLM, para cada problema en concreto.

Diseño de instrucciones

Diversos diseños de instrucciones o estrategias han sido probados con el objetivo de guiar mejor al LLM al ordenarle la extracción de la información.

En [80, 92] los autores reformulan el problema de extracción de información como un conjunto de tareas de Question Answering (QA). Una tarea de QA consiste en proveer al modelo con una pregunta de conocimiento general o de un dominio acotado y que la salida esperada sea la respuesta a dicha pregunta [27]. El argumento por el cual se elige reformular los problemas de extracción de información como problemas de QA consiste en que los modelos modernos están pre-entrenados sobre decenas de conjuntos de datos de QA, lo cual les permite resolver la tarea reformulada con mayor facilidad, sin la necesidad de entrenar o realizar fine-tuning sobre tareas de IE.

En [35], se utiliza la técnica de Chain-of-Thought (CoT) [79] para mejorar la capacidad zero-shot de extraer relaciones de los modelos. Esta técnica consiste en solicitar al LLM que explicite pasos intermedios en su razonamiento para llegar al resultado final, lo cual ha probado mejorar los resultados en tareas complejas o que requieren múltiples pasos.

En [60], se utiliza una instrucción de código y se extiende la definición de los tipos de datos a extraer con comentarios y descripciones extraídas de la guía de anotación. Los tipos de datos se definen como clases de Python, y se agrega en el *docstring* de la clase una descripción detallada del tipo de datos. A su vez, cada atributo del tipo de datos a extraer se define como un atributo de la clase, con comentarios extraidos de

Trabajo Previo	Utilización de Datos	Tipo de LLM	Tipo de instrucción	Diseño de instrucción
[35]	zero-shot	NL-LLM	LN	CoT, QA
[80]	zero-shot	NL-LLM	LN	QA
[92]	zero-shot, few-shot	NL-LLM	LN	QA, ICL
[4]	zero-shot, few-shot	Code-LLM	C	CoT, ICL
[36]	few-shot	Code-LLM	C	ICL
[44]	few-shot	Code-LLM	C	ICL
[21]	few-shot	Code-LLM	C	ICL, RAG
[75]	fine-tuning	NL-LLM	NL	-
[72]	pre-training, fine-tuning	NL-LLM	NL	-
[37]	pre-training, fine-tuning	Code-LLM	C	-
[60]	fine-tuning	Code-LLM	C	RAG
[83]	fine-tuning RL	NL-LLM	C	-
[76]	few-shot	Code-LLM	C	ICL

Tabla 2.1: Categorización de trabajos previos sobre resolución del problema de IE con modelos generativos en cuatro dimensiones: utilización de datos, tipo de LLM utilizado, tipo de instrucción utilizada y diseño de la instrucción. En la columna Tipo de instrucción, C significa que se utilizó una instrucción de código y LN una instrucción de lenguaje natural.

la guía de anotación que describen a dicho atributo.

Como se mencionó en la sección 2.1.2, el diseño de las instrucciones utilizadas es de particular importancia para obtener buenos resultados. Por eso, se debe experimentar con distintas combinaciones de estas estrategias de diseño de la instrucción como parte del proceso de *prompt engineering*.

Resumen sobre los enfoques generativos para IE

La tabla 2.1 presenta un resumen de los enfoques generativos para resolver el problema de IE sobre cuatro dimensiones: utilización de datos, tipo de LLM utilizado, tipo de instrucción y diseño de instrucción.

Se observa que la literatura está dividida entre el uso de NL-LLM y Code-LLM sin un claro consenso de qué tipo de modelo es más adecuado para esta tarea, aunque varios trabajos argumentan que los Code-LLM son mejores por su superior capacidad de generar lenguaje estructurado.

Lo mismo ocurre con el tipo de instrucción, aunque se observa que, en la mayoría de los casos, el tipo de instrucción coincide con el tipo de LLM (el uso de NL-LLM se empareja con el uso de una instrucción de lenguaje natural, y el uso de Code-LLM con el uso de una instrucción de código). El único trabajo que sale de esta línea es [83], en el cual se utiliza un NL-LLM junto con una instrucción de código. En [76], se experimenta el uso de una instrucción de lenguaje natural con un Code-LLM, y se concluye que con la cantidad suficiente de ejemplos de ICL, la instrucción de lenguaje natural puede alcanzar resultados similares a la instrucción de código. En trabajos previos no se ha explorado extensivamente el uso de un tipo de instrucción con otro tipo de LLM, lo

cual es algo interesante a analizar.

Los enfoques son bastante diversos en cuanto al uso de datos. Muchos intentan resolver el problema en modalidad zero-shot, obteniendo resultados inferiores a otros trabajos que utilizan datos para optimizar el modelo, pero aún así validando la capacidad de los LLM de generalizar a esta tarea sin una etapa de entrenamiento explícita. Otra gran proporción de trabajos intenta resolver el problema en modalidad few-shot, utilizando un volumen bajo de ejemplos para guiar al LLM en combinación con técnicas como ICL y RAG. Estos trabajos demuestran que proveer al LLM de algunos ejemplos de cómo resolver la tarea en su instrucción logra obtener buenos resultados. Finalmente, varios trabajos realizan entrenamiento del LLM en distintas etapas. Algunos trabajos realizan una etapa de pre-entrenamiento sobre corpus de código o corpus hechos a medida para lograr que el LLM mejore su capacidad de comprender y producir lenguaje estructurado. Otros trabajos realizan fine-tuning específico sobre la tarea de IE, lo cual demuestra buenos resultados, pero requiere de un volumen mayor de datos etiquetados y de buena calidad.

Finalmente, se han probado distintas estrategias para el diseño de la instrucción a utilizar con el LLM. Una de las técnicas más populares es ICL, la cual consiste en proveer al modelo de ejemplos de cómo se resuelve la tarea en la instrucción. Esta técnica se suele aplicar en conjunto con RAG para una selección más inteligente de los ejemplos a incluir en la instrucción, con el objetivo de dar ejemplos similares a la entrada sobre la cual se quiere extraer la información. Otros trabajos, en particular aquellos que intentan resolver el problema en modalidad zero-shot, modelan el problema como uno de QA, para el cual los LLM ya están ampliamente entrenados. Algunos trabajos incorporan la técnica de CoT, solicitando al LLM que explique los pasos que siguió para extraer la información, con el objetivo de mejorar los resultados en modalidad zero-shot o la capacidad del LLM de generar salidas estructuradas y anidadas.

2.3. Conjuntos de datos

La mayoría de los conjuntos de datos para extracción de información están construidos sobre corpus en inglés, lo cual limita el funcionamiento de los sistemas construidos a dicho idioma. Es reducida la cantidad de conjuntos de datos para las distintas tareas construidos sobre el idioma español.

Para la tarea de NER existen varios conjuntos de datos en español [16,19,46,61,65], siendo esta la tarea más popular en el idioma. Para RE [9, 61] y EE [16, 43, 47, 69] el volumen es menor.

Se observa que muchos de estos conjuntos de datos están compartidos con otros idiomas, lo cual reduce la cantidad de anotaciones dedicadas al idioma español. También, los dominios de dichos corpus suelen ser noticias, ciencias, farmacología o medicina.

Por otro lado, en la gran mayoría de estos conjuntos de datos, se trabaja con extracción de información a nivel de una oración y no de un documento entero. Por lo tanto, los algoritmos construidos en base a estos conjuntos de datos suelen estar entrenados para extraer información de oraciones, pero no de documentos de mayor extensión, donde la información puede estar distribuida en oraciones lejanas en el documento. A su vez, al trabajar a nivel de oraciones, se limita la cantidad de anotaciones para cada entrada del conjunto de datos.

En el alcance de este relevamiento, no se encontró ningún conjunto de datos hasta la fecha para extracción de información en español que combine las tareas de NER, RE y EE.

Dada la gran diferencia entre los dominios de los conjuntos de datos existentes y el dominio de los documentos del pasado reciente, tanto en contenido como en lenguaje

utilizado, se decidió no utilizar ninguno de esos conjuntos de datos en el trabajo actual. Sin embargo, en el futuro podría ser interesante expandir el conjunto de datos utilizado en este trabajo (ver capítulo 3) con conjuntos de datos existentes para explorar técnicas como fine-tuning.

2.4. Resolución de Correferencias

En un texto, pueden ocurrir múltiples menciones diferentes a una misma entidad (objetos físicos o eventos) del mundo real. Por ejemplo, dado el texto:

```
Juan visitó la casa donde él y su hermana crecieron.
```

Se pueden ver tres menciones, "Juan", "él" y "su", que refieren a la misma entidad, la persona Juan. Al referir las menciones a la misma entidad, se dice que estas son correferentes o que se encuentran unidas por una relación de correferencia. La identificación y resolución de correferencias es una habilidad implícita en la comprensión lectora que permite desambiguar referencias a entidades que de otra forma serían ambiguas.

La tarea de resolución de correferencias trata de identificar en el texto menciones a entidades (objetos físicos u eventos) y agrupar todas aquellas menciones correferentes a una misma entidad [27]. Resolver esta tarea, ya sea de forma implícita o explícita, es de particular importancia para que los modelos de lenguaje logren resolver otras tareas aguas abajo, como QA o IE.

Previo a intentar resolver el problema de extracción de información sobre los documentos del pasado reciente, se investigó la posibilidad de resolver el problema de resolución de correferencias en el texto como paso previo a la extracción de información, ya que se creía que resolver este paso previamente era crítico para lograr extraer información correctamente con las restricciones presentes de cantidad de datos etiquetados y calidad de los textos.

Sin embargo, luego de una corta investigación del estado del arte en cuanto a resolución de correferencias y a pruebas realizadas, se llegó a la conclusión de que la complejidad del problema reducía las probabilidades de obtener un enfoque viable, y que la mayoría de los métodos actuales requieren de grandes volúmenes de datos etiquetados para entrenar redes neuronales. En contraposición, el relevamiento del estado del arte reveló que los métodos actuales son capaces de extraer información en modalidad zero-shot o few-shot [4,21,35,36,44,76,80,92], sin la necesidad de resolver las correferencias previamente, lo cual simplifica el sistema propuesto y permite obtener una solución funcional más rápido.

Por estas razones, se decidió no continuar investigando el problema de resolución de correferencias y se cambió el foco de la investigación a intentar atacar directamente el problema de extracción de información. De todas formas, el material recabado durante esta etapa de investigación se puede encontrar en el anexo D.

2.5. Métricas de evaluación para IE

En esta sección se describen las métricas utilizadas para evaluar la calidad de los distintos modelos de extracción de información. Estas métricas son utilizadas para evaluar la calidad de las anotaciones del conjunto de datos construido (ver sección 3.4) y para evaluar los distintos experimentos realizados en la construcción del sistema de extracción de información (ver capítulo 5).

Se utilizaron las mismas métricas que en trabajos previos que utilizan modelos generativos para la extracción de información [21, 36, 41]. Las siguientes subsecciones

describen las reglas que determinan si la información extraída se considera correcta. Luego, los aciertos y errores producidos se utilizan para computar métricas agregadas.

El objetivo de utilizar las mismas métricas que en los demás trabajos es permitir comparar el rendimiento del sistema final sobre el conjunto de datos con los de trabajos previos. Sin embargo, al no existir un framework o herramienta unificada para calcular estas métricas, estas evaluaciones están sujetas a errores. A su vez, como se detalla en [52], los trabajos anteriores suelen tomar ciertas libertades para calcular estas métricas, lo cual también dificulta la comparación a través de trabajos. A pesar de esto, utilizar las mismas métricas permite realizar comparaciones aproximadas de este trabajo con otros similares.

Durante el resto de la sección se utilizan los siguientes términos:

- *Target*: para un documento, el target es el conjunto de entidades, valores, relaciones y eventos que se desea extraer. Es el *gold standard* para el documento.
- *Output*: para un documento, el output es la salida producida por un sistema de extracción de información. En un escenario ideal, el output debería ser igual al target.

2.5.1. Emparejamiento de spans

Los enfoques tradicionales para la extracción de información suelen funcionar extrayendo explícitamente los spans útiles del texto (que contienen información que se desea extraer) en base a índices de posiciones de tokens. Esto hace que muchas veces un span extraído por el algoritmo no coincida exactamente con un span del target. Por lo tanto, se suelen calcular versiones más flexibles de las métricas, donde un cierto nivel de solapamiento con el span del target se considera un *match* (una coincidencia entre un span del output y un span del target).

En los enfoques generativos, los spans útiles de texto son generados por el LLM. Por ende, tenemos problemas diferentes a los enfoques tradicionales, ya que el LLM puede generar un span útil parecido a un span del target, pero que no sea un match exacto. Por ejemplo, dado el texto:

```
La Doctora Carla Menedez integra el consejo.
```

Tenemos una entidad Persona "Doctora Carla Menedez" en el target. Un LLM puede generar una respuesta que incluya una entidad Persona con el span "Carla Menedez" o "Doctora Menedez", que ni siquiera es una sub-cadena del texto original. Para contemplar estos casos como válidos, se plantean dos estrategias para el emparejamiento de spans: Exact Match y Fuzzy Search 90. Ambas estrategias de emparejamiento se utilizan para calcular las métricas de entidades, valores, relaciones y eventos, por lo que siempre se tienen dos conjuntos de resultados.

Exact Match

En esta estrategia, dos spans son considerados una pareja válida si contienen exactamente el mismo texto.

Por ejemplo, los spans "Carla Menedez" y "Doctora Carla Menedez" no son considerados una pareja válida ya que no contienen exactamente el mismo texto.

Fuzzy Search 90

Esta estrategia utiliza la librería `thefuzz` [1], la cual implementa el algoritmo de Fuzzy String Matching para calcular un puntaje de similitud entre dos cadenas de

[1] `https://github.com/seatgeek/thefuzz`

texto. Este algoritmo utiliza la distancia Levenshtein entre dos secuencias para calcular las diferencias entre dos cadenas de texto o porciones de la cadena de texto, produciendo un puntaje entre 0 y 100 de similitud [89].

En la estrategia de emparejamiento planteada, dos spans son una pareja válida si el algoritmo retorna un puntaje mayor o igual a 90. Esto permite contemplar faltas ortográficas, desfasaje entre los spans y pequeñas alucinaciones del LLM.

Para estrategias de emparejamiento menos restrictivas, se podria reducir el umbral de aceptación del puntaje, o calcular el algoritmo de emparejamiento sobre porciones de las cadenas de texto.

2.5.2. Entidades y Valores

Una entidad o valor del output se considera correcta si existe una entidad o valor en target que cumpla lo siguiente:

- Hay un match entre el span de la entidad o valor del output y el del target.
- Ambos pertenecen a la misma clase.
- El target no es el match de otra entidad o valor del output.

2.5.3. Relaciones

Una relación del output se considera correcta si existe una relación del target que cumpla lo siguiente:

- Las entidades relacionadas por ambas coinciden; es decir, hay un match entre los spans y etiquetas de las entidades dependiente y gobernante de ambas relaciones.
- Ambas relaciones pertenecen a la misma clase.
- La relación del target no es el match de otra relación del output.

2.5.4. Disparadores de eventos

La evaluación de la extracción de eventos se separa en sus dos componentes: identificación de los disparadores e identificación de los argumentos.

Un disparador de evento del output se considera correcto si existe un disparador de evento del target que cumpla lo siguiente:

- Hay un match entre el span del disparador del output y el del target.
- Ambos pertenecen a la misma clase.
- El target no es el match de otro disparador del output.

2.5.5. Argumentos de eventos

Un argumento de un evento del output se considera correcto si se cumplen las siguientes reglas:

- El disparador del evento del output es correcto. Es decir, existe un evento en el target que se corresponde con este evento del output.
- El span del argumento en el evento del output es un match con el span del argumento en el evento del target y ambos pertenecen a la misma clase.

Para los casos en los que el valor del argumento es una lista que puede tener múltiples entidades o valores, se busca si hay un match entre todas las entidades o valores del argumento del target.

Se observa que, si un disparador de evento es extraído erróneamente, todos los argumentos asociados a ese evento se consideran un error. Esta decisión provoca que los resultados de la evaluación en cuanto a la extracción de argumentos de eventos se vean afectados negativamente por la calidad de la detección de disparadores de eventos. Sin embargo, como se argumenta en [52], este enfoque es más realista y se asemeja al tipo de funcionamiento que se espera del sistema en producción, donde los argumentos están asociados a eventos detectados por el sistema.

2.5.6. Precisión, recuperación y F1

Con las reglas descriptas en las anteriores secciones se pueden calcular los siguientes valores:

- Verdaderos positivos (VP): la pieza de información extraída es correcta. Por ejemplo, una mención a entidad que se extrae y clasifica correctamente o una relación entre dos entidades que se clasifica correctamente.
- Falsos positivos (FP): la pieza de información extraída es incorrecta. Por ejemplo, se extrajo una mención a una entidad que es incorrecta, o se extrajo correctamente la mención pero se clasifico incorrectamente en el tipo de entidad.
- Falsos negativos (FN): es una pieza de información que debería haber sido extraída pero no lo fue. Por ejemplo, una mención a una entidad que se encuentra en el target pero no dentro de la información extraída por el sistema.

Se observa que bajo esta definición de VP, FP y FN, los errores de extracción de información se penalizan tanto en los FP (se extrajo información incorrecta) como en los FN (no se extrajo la que debía ser extraída). Con estos valores, se pueden calcular el resto de las métricas estándar de clasificación.

Primero, tenemos la precisión, que se calcula como precisión $= \frac{\text{VP}}{\text{VP+FP}}$, y es una medida de que tan preciso es el sistema a la hora de extraer información. Mientras más información incorrecta se extraiga, menor será la precisión.

Segundo, tenemos la recuperación, que se calcula como recuperación $= \frac{\text{VP}}{\text{VP+FN}}$, y es una medida de cuanta información de la presente en los textos está siendo extraída correctamente. Mientras más información presente en los textos sea extraída, mayor será la recuperación.

Finalmente, tenemos la métrica F1, que es una media armónica entre la precisión y la recuperación, y se calcula como $\text{F1} = \frac{2 \times \text{precisión} \times \text{recuperación}}{\text{precisión+recuperación}}$.

2.5.7. Métricas Agregadas

En la experimentación se evalúa el sistema construido sobre las siguientes tareas:

- Extracción de entidades (o NER, por su nombre en inglés, *Named-Entity Recognition*).
- Extracción de valores, a la que llamaremos VE (por el nombre en inglés, Value Extraction).
- Extracción de relaciones (o RE, por su nombre en inglés, *Relation Extraction*).
- Extracción de disparadores de eventos (o ETD, por su nombre en inglés, *Event Trigger Detection*)

- Extracción de argumentos de eventos (o EAD, por su nombre en inglés, *Event Argument Detection*).

Las métricas mencionadas se calculan ejemplo a ejemplo. A su vez, se calculan por separado para cada tarea, y dentro de cada tarea, se pueden calcular por separado para cada clase. Por ejemplo, se calcula la precisión, recuperación y F1 para la entidad Persona en la tarea de NER.

Por lo tanto, se requiere de una forma de agregar las métricas para todos los ejemplos del conjunto de datos de evaluación. El procedimiento es el siguiente:

- Se calculan y se suman los VP, FP y FN de todos los ejemplos, separados por tarea y por clase de cada tarea. También se calcula el soporte de cada clase, definido como la cantidad de instancias de esa clase presentes en el conjunto de evaluación.
- Utilizando estos valores, se calcula la precisión, recuperación y F1 para cada clase por separado.
- Para cada tarea, se calculan dos promedios de las métricas precisión, recuperación y F1 sobre todas las clases:
  - Un promedio macro, que simplemente suma la precisión/recuperación/F1 y la divide entre la cantidad de clases de la tarea.
  - Un promedio ponderado, que pesa cada clase por el valor de su soporte.

El promedio ponderado permite que las clases menos representadas, que posiblemente tengan métricas menos representativas de la calidad de la solución, afecten en menor medida el resultado final de la evaluación.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 3

Construcción del Conjunto de Datos

Contar con un conjunto de datos etiquetado es de vital importancia para la construcción y evaluación del sistema. En el proyecto CRUZAR, trabajos previos han etiquetado datos para tareas de transcripción o extracción de entidades, pero ninguno ha etiquetado datos para las tareas de extracción de entidades, relaciones, eventos y sus argumentos de forma unificada.

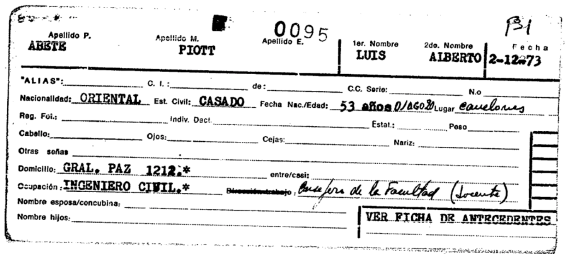
Con el fin de obtener un volumen considerable de datos y debido a la dificultad de la tarea, se decidió enmarcar la construcción de este conjunto de datos dentro de una tarea de etiquetado llevada a cabo por un equipo interdisciplinario. En este capítulo se describen los detalles del conjunto de datos, la tarea de etiquetado y los resultados obtenidos.

3.1. Corpus: fichas de la OCOA

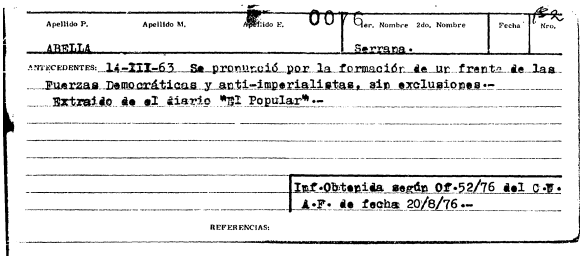
El conjunto de documentos del proyecto CRUZAR tiene un gran volumen y diversidad de documentos: fichas de organizaciones gubernamentales, reportes policiales o militares, recortes de noticias, informes y más. Estos documentos a su vez varían enormemente en formato, largo, lenguaje utilizado, tipo de escritura (manuscrita o en máquina de escribir) y calidad del escaneo. Por estas razones, se decidió concentrar este trabajo en un subconjunto de los documentos para facilitar la construcción de una prueba de concepto.

Los documentos elegidos son fichas personales gestionadas por el Organismo Coordinador de Operaciones Antisubversivas (OCOA) [1]. Estas fichas fueron completadas y mantenidas por personal del OCOA e incluían información personal de individuos de interés para el organismo. Esta información incluye características físicas, actividades, antecedentes, relaciones con grupos políticos o con otras personas de interés y más. Personas de interés típicamente podían ser aquellas con afiliaciones políticas o ideológicas de izquierda, sus contactos o familiares, o cualquiera que suscitara sospecha de subversión para el personal del organismo.

[1] `https://sitiosdememoria.uy/node/929`



(a) Cara frontal.



(b) Cara posterior.

Figura 3.1: Ejemplo de ficha completa, con su cara frontal y posterior.

3.1.1. Características de los datos

Cada ficha es un cartón impreso de ambos lados asociado a una única persona. En la cara frontal de la ficha, se encuentra información identificadora de la persona y características físicas. En la cara posterior, se encuentra un cuadro de texto escrito a mano o en máquina de escribir (a veces por distintas personas) que contiene información adicional de la persona, como relaciones laborales, actividades generales, pertenencia a organizaciones políticas, detenciones o liberaciones de centros de reclusión, traslados entre distintos lugares y más. En la figura 3.1 se pueden observar las caras frontal y posterior de una ficha de ejemplo. Se cuenta con escaneos realizados de la ficha original.

Los escaneos del archivo Berrutti pueden contener una o más caras de distintas fichas por hoja, rotadas en distintos grados o posicionadas de formas no homogéneas. En el trabajo [49] se realizó un extenso procesamiento de imágenes sobre los escaneos para segmentar las partes de ficha de una hoja, rotar cada parte de ficha para lograr una posición horizontal, y extraer información de la parte delantera de la ficha, la cual se encuentra estructurada y facilita la extracción de información por medio de algoritmos de procesamiento de imágenes. El procesamiento realizado en este trabajo permite partir de un conjunto de imágenes de fichas correspondientes a las caras traseras, segmentadas y rotadas correctamente.

Contar con un conjunto de imágenes de caras posteriores de ficha es una de las principales razones por las cuales se eligió este subconjunto de datos para la realización de este trabajo. Otras razones que justifican la selección de las fichas de la OCOA son las siguientes:

- Todas las fichas siguen un formato y lenguaje similar, dado que fueron escritas por personal del mismo organismo. Esto permite obtener un conjunto homogéneo de datos para explorar.
- El texto de las fichas es denso en entidades, relaciones y eventos de interés para

investigadores: personas, organizaciones, lugares, fechas, relaciones entre personas, eventos de detención y movimientos de personas entre centros de detención y más. Esto hace que sea un conjunto rico de analizar por un sistema de extracción de información.

- Estas fichas se encuentran identificadas y filtradas del resto de los documentos del proyecto CRUZAR, lo cual ahorró esfuerzo en la etapa de selección de datos.

El preprocesamiento de las imágenes realizado en [49] resultó de gran utilidad para el presente trabajo, permitiendo ahorrar etapas de preprocesamiento que hubieran hecho de este un trabajo mucho más extenso.

3.1.2. Preprocesamiento

Si bien el trabajo [49] logró segmentar y posicionar correctamente las caras frontales y posteriores de las fichas, el resto del procesamiento y extracción de datos de dicho trabajo se concentró en la cara frontal de la ficha, la cual se encuentra estructurada en un formato de tabla y permite extraer cierta información utilizando técnicas clásicas de procesamiento de imágenes.

La tarea de etiquetado de las partes posteriores de la ficha requiere previamente obtener el texto plano de la ficha a partir de su imagen. Para lograr esto, fueron necesarios algunos pasos adicionales de transcripción y posprocesamiento de los textos transcriptos.

En primer lugar, se debió procesar la imagen con un modelo de *Optical Character Recognition* (OCR) para transcribir el texto de la ficha. Con el objetivo de obtener transcripciones de buena calidad, se realizaron varios experimentos con el modelo MiniCPM-Llama3-v2.5 [87]. Se experimentó con el modelo sin ninguna intervención y con versiones optimizadas mediante fine-tuning. Para esto, en primera instancia, se creó un conjunto de datos de 80 fichas transcriptas manualmente (50 para entrenamiento y 30 para evaluación) y se realizó fine-tuning del modelo sobre este conjunto. Al obtener resultados prometedores con este experimento, se realizó una etapa de fine-tuning más extensa con 174 fichas para entrenamiento y 40 para evaluación transcriptas manualmente. Este conjunto de datos más extenso no se intersecta con el conjunto de entrenamiento de 50 fichas de la etapa previa, y se utiliza luego para comparar todas las variantes de modelos probadas. La versión del modelo obtenida en la segunda etapa de fine-tuning, entrenada utilizando el conjunto más extenso de datos, obtiene un puntaje de 2.55 de Word Error Rate (WER) promedio sobre el conjunto de datos de evaluación. Esta métrica evalúa la cantidad de errores en la transcripción (inserciones, eliminaciones o substituciones) a nivel de palabra comparando la salida del modelo con la transcripción deseada, y un puntaje de 2.55 significa que en promedio se cometen entre 2 y 3 errores por transcripción. Una descripción más detallada de la obtención de las transcripciones y la experimentación realizada se puede encontrar en el anexo A

Debido a las características de las imágenes, las transcripciones no están libres de errores. Dado que el foco de este trabajo no es mejorar la calidad de la transcripción, se optó por seleccionar manualmente las transcripciones de mayor calidad para el conjunto de datos. Este proceso manual consistió en la selección y corrección de aproximadamente 1000 documentos, de largos y formatos variados, para la etapa de etiquetado de información. Gracias a los esfuerzos realizados por obtener una transcripción de buena calidad en primera instancia, este proceso de selección y corrección tomó aproximadamente unas 10 horas.

Este conjunto de aproximadamente 1000 documentos corregidos, junto con la salida original del OCR, se podría utilizar como conjunto de datos para entrenar un modelo de lenguaje capaz de corregir los errores de la salida del OCR, mejorando aún más el

proceso de transcripción. Este conjunto de datos, si bien no tiene una relación directa con el problema de extracción de información, es también un aporte relevante de esta tesis.

Aprovechando la capacidad del modelo MiniCPM-Llama3-v2.5 no solo de transcribir el texto sino que también de hacerlo de forma estructurada (ver anexo A), en la etapa de fine-tuning se entrena el modelo para que extraiga la información de la ficha separada en las secciones Causante, Nro. Ficha, Fecha, Texto y Referencias, como se puede ver en el ejemplo 3.1. El contenido de estas secciones se puede observar en la figura 3.1: el nombre y apellido del causante se encuentran en la esquina superior izquierda; el número de ficha es un número de tres o cuatro dígitos escrito a máquina que suele aparecer superpuesto en la parte superior de la ficha; la fecha aparece en el recuadro "Fecha" de la esquina superior derecha; el texto se encuentra en el cuerpo central de la ficha; las referencias se encuentran en el recuadro inferior derecho de la ficha. No todas las fichas siguen este exacto mismo formato, e incluso las que lo siguen suelen tener variaciones, como por ejemplo, que el nombre de la persona se salga del recuadro superior izquierdo o que el texto abarque también el recuadro inferior derecho de la ficha.

Ejemplo 3.1: Ejemplo de transcripción obtenida por el OCR, con las secciones Causante, Nro. Ficha, Fecha, Texto y Referencias.

```
Causante: ANA RODRIGUEZ

Nro. Ficha: 4834

Fecha: 11/07/77

Texto: ANTECEDENTES: Junto con su esposo fue detenida por
hallarse en su domicilio materia de izquierda
Recupera su libertad 21.12.76 segun B.de N.356.-

Referencias: Inf. Obt. segun Novedades de DNII de 20.12.76
No. 355.-
```

Las secciones de la estructura son las siguientes:

- Causante: este es el nombre de la persona objeto de la ficha, y sobre la cual se detalla más información en la sección Texto.

  En las imágenes originales de las fichas, el nombre de la persona se encuentra escrito con máquina, a veces de forma parcial o ilegible. Por lo tanto, no siempre este nombre se encuentra completo; es posible que esta sección solo tenga el nombre de pila de la persona o el apellido. También es muy común que dichos nombres tengan faltas ortográficas.

  En algunos casos, no se logró extraer ninguna parte del nombre de la persona de la imagen. Para mantener consistencia con las demás fichas y facilitar la tarea de etiquetado, en esos casos se relleno dicha sección con el nombre artificial "Fulano Mengano". Crear esta entidad ficticia facilita enlazarla con otras entidades en relaciones o eventos.
- Nro. Ficha: este número identifica a la ficha dentro del archivo mantenido por la OCOA y es único para cada persona dentro del fichero. A efectos de la tarea de etiquetado, este identificador no tiene importancia.
- Fecha: esta fecha puede significar la primera fecha en la cual se rellenó la ficha, aunque la ficha puede tener información agregada en distintas fechas y por distintas personas. Por ende, esta fecha resulta ambigua. A efectos prácticos, esta

fecha no tiene relevancia en la tarea de etiquetado, pero de encontrarse igual se etiqueta como un valor de Tiempo.

- Texto: esta es la sección más importante para la tarea de etiquetado y en la que se concentran la mayor cantidad de entidades, relaciones y eventos que se deben etiquetar.

  Esta sección contiene información sobre el causante: datos personales (contactos, familiares, trabajo, participaciones en eventos o reuniones), pertenencia a organizaciones o listas electorales, detenciones, liberación de centros de detención, traslados (dentro o fuera del país) y más. Esta información puede haber sido completada en diversos puntos del tiempo y por distintas personas.

- Referencias: En esta sección se mencionan archivos o partes de información utilizados para extraer la información sobre el causante que se encuentra en la ficha.

3.2. Ontología de dominio

El objetivo de este trabajo es extraer eventos, relaciones y entidades de interés de los documentos. En el futuro, se planea alimentar un grafo de conocimiento utilizando dicha información, con el objetivo de habilitar nuevos análisis sobre los archivos. Con estos objetivos en mente, se define una ontología de dominio que cumple dos propósitos: guiar el proceso de extracción de información llevado a cabo en este trabajo y actuar como esquema conceptual para organizar el grafo de conocimiento que se construirá en el futuro. El resto de esta sección se encarga de describir la ontología de dominio construida.

El diagrama UML de la figura 3.2 presenta los objetos y propiedades a extraer del texto siguiendo el formato Chowlk [11].

En las siguientes secciones, se incluye una breve descripción de la información a extraer. Para mayor detalle sobre las reglas de etiquetado de esta información, ejemplos y casos excepcionales, se sugiere leer la guía de etiquetado mencionada en 3.3.1.

A continuación, se presentan las clases, relaciones y atributos relevantes definidos en la ontología. Todos los ejemplos que se presentan en esta sección son ficticios y se incluyen únicamente a modo de explicación.

3.2.1. Entidades

Una entidad es un objeto o conjunto de objetos del mundo real. Una mención es una referencia a una entidad, y en el texto puede haber varias menciones a la misma entidad; el objetivo es etiquetarlas todas. Las menciones a la misma entidad luego se deben enlazar por una relación de Correferencia, como se explica en 3.2.3.

Las entidades pueden ser referenciadas en el texto por su nombre, indicado por un nombre propio, una frase nominal, una sigla, un pronombre, un rol, un título y más. Por ejemplo, las siguientes son menciones a entidades: “José Perez”, “él”, “ellos”, “la organización”, “el chofer”, “la profesora de inglés”, “M.L.N”, “el esposo”, “el recluso nro. 123 del E.M.R.1”, “Av. Brasil 1234”.

Las entidades a su vez pueden verse modificadas por adjetivos (“el sedicioso Pérez”), encontrarse anidadas (“El director de la USBC”) o mencionar de forma conjunta a varias entidades (“los prófugos María Martínez, José Gómez y Carla Rodríguez”).

En esta primera versión del conjunto de datos el objetivo es extraer tres clases de entidades: `Persona`, `Organización` y `Lugar`.

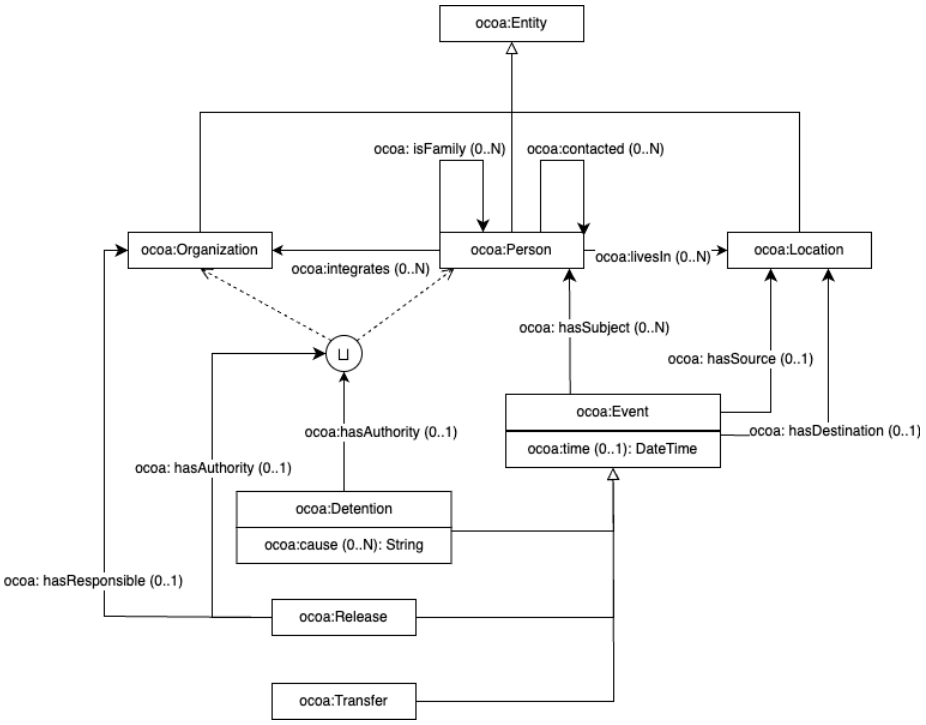


Figura 3.2: Ontología de la información a extraer, siguiendo el formato Chowlk [11].

`Persona`

Toda persona o conjunto de personas (que no sea una organización per se; por ejemplo: "los agitadores", "los protestantes") refiere a una entidad de tipo `Persona`. En estos textos, las personas son mencionadas como: causantes, participantes de organizaciones, jueces, miembros del ejército o la fuerza militar y más. Las personas pueden ser referidas por su nombre ("CAGGIANI María"), un alias o apodo ("Gordo"), pronombres ("él", "ella"), una ocupación o rol ("el delincuente", "el juez", "la prima"), etc.

`Organización`

Toda organización o conjunto de organizaciones refiere a una entidad de tipo `Organización`. Ejemplos de organizaciones que se pueden encontrar fácilmente en estos textos son organizaciones gubernamentales o militares ("D.N.I.I", "OCOA", "G.A.1", "J.M.I"), políticas ("Comité Ejecutivo Departamental del Partido Colorado", "Frente Amplio"), las llamadas subversivas ("M.L.N", "la orga", "la organización", "Col. 23"), instituciones ("Liceo Zorrilla", "Facultad de Medicina"), empresas ("Textil FIBRATY", "CUTCSA"), listas políticas ("lista 1001 del Partido Colorado a las elecciones del 10/11/78"), sindicatos, periódicos, clubes deportivos, bandas musicales y más. Subáreas de una organización se pueden mencionar de forma anidada ("Comité directivo de la UJC"), y también deben ser anotadas como una entidad de tipo `Organización`.

`Lugar`

Menciones a lugares geográficos, que no refieren directamente a una organización, refieren a una entidad de tipo `Lugar`. Estos pueden ser direcciones ("Gonzalo Ramírez 1921", "Ruta 3 KM 301"), locales ("la sede del CNT"), referencias a domicilios ("su domicilio", "la casa de sus padres", "el apartamento"), barrios/ciudades/departamentos/países ("Cerro", "Paysandú", "Departamento de Maldonado", "el país", "Argentina"), cárceles ("E.M.R.1", "Punta Carretas"), instalaciones ("la fábrica de Alpargatas", "Aeropuerto Internacional de Carrasco") y más.

En ciertas ocasiones puede haber ambigüedad en cuanto a si una mención es un `Lugar` o una `Organización`. Por ejemplo, la mención "la fábrica de Alpargatas" refiere a un `Lugar` en el fragmento "fue detenido en la fábrica de Alpargatas", pero refiere a una `Organización` en el fragmento "es el gerente de la fábrica de Alpargatas". En estos casos, la ambigüedad debe resolverse analizando el contexto que acompaña la mención.

3.2.2. Valores

Un Valor es una cadena de texto que provee mayor información sobre una Entidad o Evento. Como se verá más adelante en las secciones Relaciones y Eventos, los valores se pueden asociar a una Entidad mediante Relaciones o a un Evento mediante argumentos.

En esta primera versión del conjunto de datos el objetivo es extraer dos valores: `Tiempo` y `Causa`.

`Tiempo`

Este valor refiere a cualquier cadena de texto que referencie un punto en el tiempo o un lapso. Algunos ejemplos de esto son fechas ("11/nov/73", "1/XI/1968", "12 de abril del 1976"), días ("el 1ro", "Viernes"), meses, años, referencias relativas ("ayer", "mañana", "el pasado", "posteriormente"), lapsos ("entre el 28 y 30 de noviembre") y más.

`Causa`

Este valor refiere a cualquier cadena de texto que referencie una causa por la cual una persona puede haber sido detenida. Algunos ejemplos de esto son actividades consideradas subversivas en la época ("participó en volanteadas", "participó en manifestación no autorizada", "tener material de izquierda"), violación de las leyes ("rapiña de vehículo", "ASOC. PARA DELINQUIR", "DESACATO", "al amparo de las MPS"), pertenecer a grupos subversivos ("pertenecer al M.L.N") y más.

El objetivo de este valor es brindar más información sobre el evento `Detención`, descrito más adelante.

3.2.3. Relaciones

Una relación es un par de entidades o una entidad y un valor que tienen un significado semántico según el texto. Por ejemplo, la relación `domicilio` une una entidad `Persona p` con una entidad `Lugar l`, cuando `p` se domicilia en `l`. En este caso, la relación se puede representar con la terna ordenada `(p, domicilio, l)`.

Las relaciones son ternas ordenadas, por lo cual la terna `(a, relación_x, b)` no es lo mismo que la terna `(b, relación_x, a)`. A su vez, las relaciones tienen dominio y recorrido especificados, por lo cual solo pueden unir entidades de los tipos indicados.

Las demás reglas de anotado de las relaciones se pueden encontrar en la guía de anotación.

En esta primera versión del conjunto de datos, el objetivo es anotar las siguientes relaciones: `integra`, `domicilio`, `contacto`, `familiar` y `correferencia`.

`integra`

Esta relación une una entidad de tipo `Persona` con una entidad de tipo `Organización` y se usa cuando la persona integra una organización. Esta relación puede presentarse cuando la persona posee un puesto de trabajo en una organización, o integra, milita, colabora en una.

`domicilio`

Esta relación une una entidad de tipo `Persona` con una entidad de tipo `Lugar` y se usa cuando la persona se domicilia formalmente en un lugar.

`contacto`

Esta relación une dos entidades de tipo `Persona`. Se utiliza cuando en el texto hay clara evidencia de que dos personas tuvieron contacto de algún tipo: hablaron (por medio de cartas o presencialmente), se encontraron o participaron juntas de alguna acción o evento.

`familiar`

Esta relación une dos entidades de tipo `Persona`. Se utiliza cuando ambas personas están relacionadas por un vínculo familiar directo (padre, madre, hijo/a, hermano/a, primo/a, etc.) o político (esposo/a, pareja, cuñado/a, suegro/a, etc.).

`correferencia`

Esta relación es particular y se diferencia de las demás en que no une a dos entidades distintas bajo un concepto semántico. Esta relación se utiliza para unir dos menciones a la misma entidad. Es decir, si en el texto se anotan varias menciones a una entidad, que representan la misma entidad del mundo real, todas esas menciones deberían estar unidas por una cadena de relaciones de correferencia.

En el ejemplo 3.2 se muestra un fragmento de texto donde se encuentran distintas menciones a la misma entidad. En este fragmento se tienen tres menciones a entidades `Persona`, pero dicha entidad es la misma en el mundo real ya que "Fulano Mengano", "El mismo" y "el argentino" refieren a exactamente la misma persona. Por ende, estas tres menciones son correferentes.

Ejemplo 3.2: Fragmento de texto donde fueron señaladas todas las menciones a una misma entidad.

```
Causante: [Fulano Mengano]
Texto: [El mismo] es argentino terrorista. El dia 26/07/68
[el argentino] fue detenido en la frontera.
```

Como se observó en el estado del arte y en experimentos realizados (ver anexo D), extraer esta relación de correferencia de forma explícita es muy difícil para modelos generativos, y para modelos que siguen un enfoque tradicional se requiere de entrenamiento supervisado para lograr buenos resultados. Por ende, en el diseño del sistema de extracción de información planteado en este trabajo se decidió no extraer de forma

Argumento	Tipo	Descripción
`sujeto`	Lista de `Persona`	Persona o grupo de personas que fueron detenidos.
`autoridad`	`Organización` o `Persona`	Organización (de índole militar o policial) que realizó la detención. También puede referirse a un grupo de personas o persona que pertenecen a una `Organización`.
`causa`	Lista de `Causa`	Causa por la cual se realizó la detención.
`fecha`	`Tiempo`	Fecha y/o tiempo de la detención.
`origen`	`Lugar`	Lugar donde fue detenida la persona o grupo de personas.
`destino`	`Lugar`	Lugar a donde fueron trasladadas las personas inmediatamente luego de ser detenidas.

Tabla 3.1: Argumentos del evento `Detención`.

explícita la información de correferencias. Sin embargo, se observa que los modelos generativos son capaces de resolver enlaces de correferencia de forma implícita. Por ejemplo, son capaces de reconocer que "Fulano Mengano" es un "argentino terrorista" y que "fue detenido en la frontera", como se observa en la sección 5.3.

Varios trabajos del estado del arte utilizan información de correferencias entre menciones para intentar mejorar la extracción de entidades, relaciones y eventos [67, 71]. Por lo tanto, si bien esta información no es utilizada por el sistema desarrollado en el presente trabajo, se decidió anotar los enlaces de correferencia dado que contar con esta información puede facilitar la exploración de nuevos métodos en futuros trabajos.

3.2.4. Eventos

Un evento es algo que ocurre en el tiempo y que provoca un cambio en el estado de las cosas. Un evento tiene dos componentes importantes: un disparador y los argumentos del evento.

El disparador es una palabra o cadena de palabras que claramente expresa la ocurrencia de un evento. Es la componente principal del evento y de la cual se desprenden los argumentos.

Los argumentos del evento ayudan a describir el evento en mayor detalle. Pueden ser los participantes del evento, así como el momento y lugar de ocurrencia u otras características propias de cada tipo de evento. En esta tarea, los argumentos se representan como relaciones entre el disparador y entidades o valores anotados en el texto. Muchas de las reglas aplicadas a relaciones también aplican a los argumentos.

En esta primera versión del proyecto de anotación interesa anotar tres tipos de eventos: `Detención`, `Liberación` y `Traslado`.

`Detención`

El evento `Detención` ocurre cuando una persona o grupo de personas son detenidas por las fuerzas militares o policiales. Los argumentos de este evento se detallan en la tabla 3.1.

Argumento	Tipo	Descripción
`sujeto`	Lista de `Persona`	Persona o grupo de personas que fueron liberadas de la detención.
`autoridad`	`Organización` o `Persona`	Organización o Persona (en el caso de un Juez) que dictaminó la libertad del detenido.
`fecha`	`Tiempo`	Fecha y/o tiempo de la liberación.
`origen`	`Lugar`	Lugar de donde fue liberada la persona, típicamente un establecimiento de reclusión.
`destino`	`Lugar`	Lugar que indica a donde fue llevada la persona luego de ser liberada. Esta información es útil en casos en que una persona es liberada de un centro de reclusión para ser llevada a otro.
`responsable`	`Organización`	Organización que está encargada de monitorear/vigilar a la persona liberada en casos de "Libertad Vigilada" ("L.V.")

Tabla 3.2: Argumentos del evento `Liberación`.

Argumento	Tipo	Descripción
`sujeto`	Lista de `Persona`	Persona o grupo de personas que se trasladaron.
`fecha`	`Tiempo`	Fecha y/o tiempo del traslado.
`origen`	`Lugar`	Lugar de origen del traslado.
`destino`	`Lugar`	Lugar de destino del traslado.

Tabla 3.3: Argumentos del evento `Traslado`.

`Liberación`

El evento `Liberación` ocurre cuando una persona o grupo de personas son liberadas de la detención o encarcelamiento. Los argumentos de este evento se detallan en la tabla 3.2.

`Traslado`

El evento `Traslado` ocurre cuando una persona o grupo de personas se trasladan desde un lugar a otro, ya sea voluntariamente o forzosamente. Se puede dar en casos de viajes, emigraciones o inmigraciones, huidas, mudanzas o más. Este evento no se utiliza en casos de visitas o concurrencias a lugares por un lapso corto de tiempo. Por ejemplo, si una persona concurre a un centro de detención a visitar a otra, o a un lugar para una reunión o manifestación, no se considera un evento `Traslado`. Los argumentos de este evento se detallan en la tabla 3.3.

3.3. Tarea de etiquetado

Dado el volumen de los datos y la complejidad de su etiquetado, se diseñó una tarea de etiquetado a llevarse a cabo por un equipo de etiquetadores. El equipo estuvo conformado por tres estudiantes de Ciencias Sociales y un estudiante de Ingeniería en Computación, y coordinado por el autor de este trabajo, quien también etiquetó parte de los datos.

La tarea de etiquetado contó con las siguientes etapas:

- Introducción: en esta etapa se introdujo a los participantes a la tarea, los datos y la herramienta a utilizar para etiquetar los datos. También, se introdujo a los participantes a la guía de etiquetado.
- Práctica: se realizó una iteración de etiquetado de prueba para que pudieran plantear dudas o críticas, y se pudiera refinar la configuración de la tarea y la guía de etiquetado.
- Etiquetado: en esta etapa se realizó el etiquetado oficial de los documentos, con controles cada 15 días.
- Evaluación: en esta etapa se calcularon métricas de acuerdo entre los anotadores y se obtuvieron estadísticas sobre el conjunto de datos.
- Curado: en esta etapa final, se curó un conjunto de los datos etiquetados para corregir posibles errores y homogeneizar el conjunto, produciendo la versión final del conjunto de datos.

3.3.1. Guía de etiquetado

Con el fin de asistir a los etiquetadores, se elaboró una guía de anotación que incluye contexto de la tarea y los datos, especificaciones para cada uno de los tipos de información a extraer, las reglas de la tarea de etiquetado, ejemplos y recomendaciones.

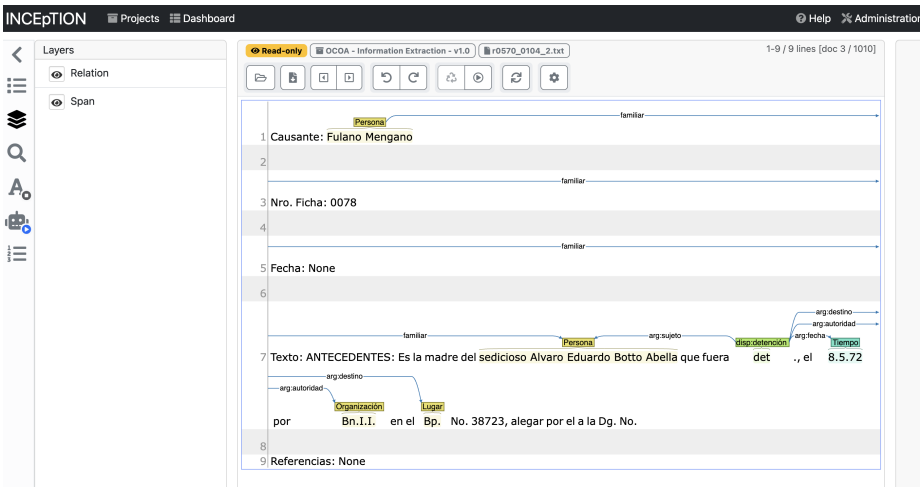
La elaboración de la guía de etiquetado, el diseño de las reglas y la documentación de los casos borde están fuertemente influenciados por las guías de etiquetado del conjunto de datos ACE05 [13, 14]. ACE05 es un conjunto de datos multilingüe de extracción de información que contiene anotaciones para entidades, relaciones y eventos en inglés, chino y árabe. Los textos incluidos dentro de ACE05 corresponden a noticias, conversaciones en medios, blogs, foros de discusión y conversaciones telefónicas. Si bien los lenguajes y la naturaleza del conjunto distan de la tarea de etiquetado de este trabajo, las guías de anotación se encuentran completas y mucha de su información y reglas son transferibles a la tarea en cuestión.

Dada la extensión de dicha guía, se incluye como documento anexo.

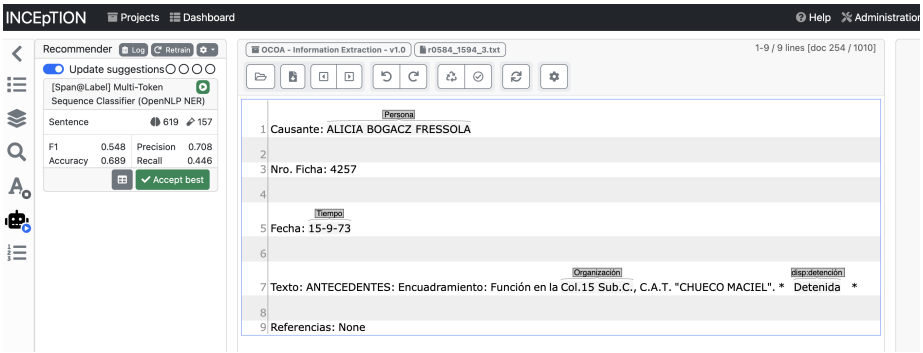
3.3.2. Herramienta para la anotación colaborativa

Para el etiquetado del conjunto de datos se utilizó la herramienta INCEpTION [31]. Gracias a los esfuerzos de los integrantes del grupo CRUZAR, una instancia privada de la herramienta fue desplegada en los servidores del grupo. La herramienta ya había sido probada por el grupo en otros proyectos y utilizada en otros esfuerzos de etiquetado [10].

La figura 3.3 contiene capturas de pantalla de la herramienta donde se pueden ver algunas de las funcionalidades básicas. En la primera captura, se puede ver la vista de un documento junto con sus anotaciones. Se pueden visualizar los spans de texto anotados como entidades, valores o disparadores de eventos y los enlaces entre estos spans que representan relaciones o argumentos del evento.



(a) Vista general de un documento anotado.



(b) Vista de un documento con sugerencias de anotaciones.

Figura 3.3: Ejemplos de vistas de un documento a anotar en la herramienta INCEpTION.

La herramienta cuenta con una funcionalidad de etiquetado asistido que resultó útil para acelerar el proceso. Esta funcionalidad utiliza algoritmos de aprendizaje automático para aprender los patrones de etiquetado del anotador para algunas de las clases más sencillas (por ejemplo, valores o entidades) y sugiere potenciales entidades o valores en el texto, los cuales el anotador debe validar y corregir. Esta funcionalidad, aunque suele cometer errores, acelera la creación de muchas de las etiquetas sencillas, permitiendo que el anotador dedique la mayor parte del esfuerzo al anotado de tareas más complicadas como la anotación de relaciones o eventos. En la imagen inferior de la figura 3.3 se puede observar un ejemplo de las sugerencias producidas por la herramienta para un ejemplo en concreto.

INCEpTION también cuenta con facilidades para la gestión de la tarea de etiquetado, el curado de las etiquetas y otras funcionalidades más.

3.3.3. Posprocesamiento de las anotaciones

Las etiquetas generadas con la herramienta INCEpTION se exportan en el formato UIMA CAS XMI [2]. Es necesario aplicar varios pasos de posprocesamiento para poder llevar las anotaciones desde ese formato hacia el tipo de datos con el cual trabaja el sistema de extracción de información diseñado y las herramientas de evaluación construidas. Dicho tipo de datos se rige por las estructuras definidas en la sección 4.2.

Previo al posprocesamiento, se realizan validaciones automáticas para verificar que las anotaciones cumplen todas las reglas de tipos y estructura especificadas en la guía de anotación. Las anotaciones que no cumplen con las especificaciones son corregidas manualmente.

El primer paso de posprocesamiento consiste en traducir las anotaciones del formato UIMA CAS XMI a instancias de los tipos de datos definidos en la sección 4.2. Para esto, se construyó un script [3] que automáticamente traduce las anotaciones haciendo uso de la herramienta DKPro cassis [4], la cual es una librería de Python que permite cargar y navegar de forma sencilla las anotaciones en formato UIMA CAS XMI.

El segundo paso de posprocesamiento es la resolución de los enlaces de correferencia. Como se mencionó en la sección 3.2.3, durante la tarea de etiquetado se anotan todas las menciones a una misma entidad, y se las enlaza con una relación de correferencia. Si bien los enlaces de correferencia no son utilizados en la construcción del sistema de información de extracción del presente trabajo, estos enlaces pueden ser útiles para explorar otros métodos en trabajos futuros.

Se definió entonces que uno de los objetivos del sistema de extracción de información a desarrollar es extraer una única mención por entidad del texto y que se deben ignorar los enlaces de correferencia. Por lo tanto, es necesario aplicar un paso de posprocesamiento al documento obtenido de la tarea de anotación para resolver los enlaces de correferencia, eliminarlos y obtener una única mención por entidad que actúe como representante de la entidad.

Utilizando los enlaces de correferencia, se pueden identificar todas las menciones a la misma entidad y, por ende, todas las relaciones y eventos de los que participa esa entidad. Como mención representativa se utiliza la primera mención en el texto, la cual típicamente es el nombre propio de la entidad o su versión más extensa. La decisión de extraer únicamente la primera mención a la entidad y no todas hace que la tarea de extracción de información sea más simple y reduce la cantidad de información a extraer.

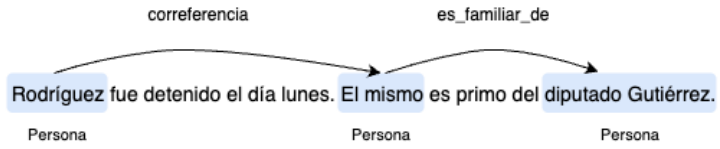
El ejemplo de la figura 3.4 ilustra la resolución de los enlaces de correferencia. En la primera imagen se puede observar el documento anotado antes de aplicar el paso de resolución de enlaces de correferencia, tal como se espera que el mismo sea anotado en la tarea de etiquetado. Se puede observar que existen dos spans de texto anotados con la etiqueta `Persona`, "Rodríguez" y "El mismo"; ambos son menciones a la misma entidad del mundo real, la persona de apellido Rodríguez, y por lo tanto se encuentran unidos por un enlace de correferencia. A su vez, la mención "El mismo" está unida por una relación `es_familiar_de` con otra entidad.

En la segunda imagen se puede observar el documento anotado luego de aplicar el paso de resolución de enlaces de correferencia. Todas las menciones a la entidad del mundo real Rodríguez se unifican en una única mención representante (la primera en aparecer en el texto, "Rodríguez"). Se observa que la mención representante pasa a ser la partícipe de las relaciones y eventos en los cuales participaban las menciones

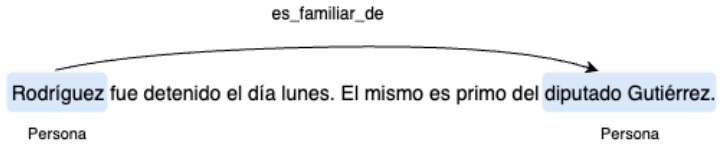
[2] `https://uima.apache.org/`

[3] `https://gitlab.fing.edu.uy/mh/information_extraction/-/blob/main/information_extraction/dataset/dataset.py?ref_type=heads`

[4] `https://github.com/dkpro/dkpro-cassis`



(a) Documento anotado antes de aplicar el paso de resolución de enlaces de correferencia.



(b) Documento anotado luego de aplicar el paso de resolución de enlaces de correferencia.

Figura 3.4: Ejemplo de documento anotado antes y después de aplicar el paso de resolución de enlaces de correferencia. Se puede observar que luego de aplicar el paso, de las menciones correferentes solo permanece la que primero aparece en el texto. A su vez, esta mención pasa a participar de todas las relaciones y eventos de los cuales participaban las otras menciones correferentes.

correferentes; en este caso, la mención "Rodríguez" pasa a participar de la relación `es_familiar_de` con la otra entidad del documento.

3.4. Resultados de la tarea de anotación

En esta sección se detallan los resultados de los esfuerzos de anotación en términos de volumen y acuerdo entre anotadores. Luego, se detalla el proceso de curado y la creación del conjunto final de datos.

3.4.1. Cantidad de documentos anotados

Como se mencionó previamente, para la tarea de anotación se seleccionaron y curaron 1000 documentos. El objetivo de la tarea fue anotar la mayor cantidad de documentos posibles, con la mayor calidad, en el tiempo limitado del cual se disponía de los anotadores.

Cada anotador recibió un conjunto aleatorio de documentos a anotar, solapando estos conjuntos entre distintos anotadores para poder calcular métricas de acuerdo entre ellos y facilitar la tarea de curado de los datos.

En total, se obtuvieron 515 documentos diferentes anotados por al menos un anotador y 125 documentos anotados por al menos dos anotadores, los cuales se utilizaron para calcular métricas de acuerdo.

La tabla 3.4 muestra la cantidad de documentos, cantidad de anotaciones y cantidad de anotaciones inválidas por anotador. El anotador "Autor" refiere al autor del presente trabajo, quien definió la tarea de etiquetado y construyó la guía de etiquetado. Los anotadores 1-4 son los estudiantes que participaron en la tarea de etiquetado. Se consideran inválidas aquellas anotaciones que no cumplen las reglas o estructuras planteadas en la guía de anotación, y que deben ser corregidas manualmente. Esta validación se ejecuta de forma automática a la hora de posprocesar el conjunto de datos etiquetado. De esta columna se puede intuir que hay lugar para mejora en cuanto

Anotador	# documentos	# anotaciones	# anotaciones inválidas
Autor	274	3141	0
Anotador 1	216	2517	16
Anotador 2	79	838	7
Anotador 3	54	469	76
Anotador 4	22	222	8

Tabla 3.4: Cantidad de documentos, cantidad de anotaciones y cantidad de anotaciones inválidas por anotador. El anotador "Autor" es el autor del presente trabajo y los anotadores 1-4 son los estudiantes que participaron de la tarea de etiquetado. La cantidad de anotaciones se calcula como la suma de entidades, valores, relaciones, disparadores y argumentos anotados.

al control de la validez de las anotaciones, ya que hay un anotador con un número considerablemente alto de errores. Por limitaciones de la herramienta utilizada, no se pudieron aplicar validaciones de tipo y de cumplimiento de las reglas al momento de anotar, lo cual podría ser útil para evitar estos tipos de errores una vez avanzada la tarea de etiquetado. Se puede observar también que el esfuerzo fue desbalanceado y que, en algunos casos, la cantidad de documentos anotados fue considerablemente menor.

3.4.2. Evaluación de la calidad de las anotaciones

En las siguientes secciones se evalúa la calidad de las anotaciones siguiendo dos estrategias:

- Acuerdo con autor: se comparan las anotaciones de cada anotador contra las del autor. El autor es quien define la tarea de etiquetado e idealmente debería tener el mayor conocimiento para la tarea. Siguiendo esta estrategia, se puede evaluar la calidad de las anotaciones contra las de un anotador experto.
- Acuerdo entre anotadores: se comparan las anotaciones de los anotadores entre si. Siguiendo esta estrategia, se puede evaluar la homogeneidad de las anotaciones de los distintos anotadores.

Evaluación de la calidad según acuerdo con autor

Siguiendo las recomendaciones planteadas en [23], se mide el acuerdo entre anotadores utilizando las mismas métricas para evaluar la calidad de los sistemas de extracción de información (ver sección 2.5). Para esto, se toman pares de anotadores y se mide el acuerdo entre ambos considerando las anotaciones de uno como el *gold standard*. Como representante del acuerdo entre anotadores se utiliza la métrica F1, la cual es independiente de qué anotador del par se considere como *gold standard*. Luego, el acuerdo global entre anotadores se puede medir como el promedio entre las métricas F1 de cada par de anotadores. También se presentan las métricas de precisión y recuperación, utilizadas para calcular la métrica F1, para un análisis más detallado de los resultados obtenidos para las distintas tareas.

Una primera forma de medir la homogeneidad y calidad de las anotaciones es midiendo el acuerdo entre los anotadores independientes y el autor. Este acuerdo tiene particular importancia, dado que los autores son quienes definen la tarea y las reglas de etiquetado.

Aunque muchos conjuntos de datos de PLN no hacen públicas o de fácil acceso las métricas de acuerdo entre anotadores, podemos tomar como referencia el conjunto de

datos OntoNotes 2005 [22], que luego de varios años de experiencia en el anotado de datos y su utilización para entrenar algoritmos de aprendizaje automático, estableció un mínimo de 90 % de acuerdo entre anotadores para cada tarea. Todas las anotaciones que no superan ese umbral son pasadas a revisión por expertos y mejoradas hasta lograr alcanzar el mínimo global para la tarea en todas las anotaciones.

Las tablas 3.5, 3.6 y 3.7 presentan los valores obtenidos para las métricas de acuerdo entre los anotadores independientes y el autor, para cada una de las tareas: extracción de entidades (NER), valores (VE), relaciones (RE), y disparadores y argumentos de eventos (ETD y EAD). Se evalúa el acuerdo en base a las métricas precisión, recuperación y F1, utilizando la estrategia de emparejamiento Fuzzy (ver subsección 2.5.1), considerando al autor como el *gold standard*.

Observando la tabla 3.5, las tareas para las cuales el promedio de precisión es más bajo son las tareas RE (44.7) y EAD (48.5). Un valor bajo de precisión quiere decir que se produjeron múltiples etiquetas incorrectas. Ambas tareas comparten características similares: implican el enlazado de un par de spans y la clasificación del enlace.

Según la métrica de recuperación (ver tabla 3.6), las tareas con un promedio de recuperación más bajo también son RE (22.9) y EAD (42.6), aunque para la tarea RE el valor promedio de recuperación es bastante más bajo que el valor promedio de precisión (44.7). Un valor bajo de recuperación quiere decir que muchas de las etiquetas producidas por el autor no fueron reconocidas por el anotador.

Para las demás tareas (NER, VE y ETD) también se observa que los valores promedio de precisión suelen ser más altos que los valores promedio de recuperación, siendo más significativa esta diferencia para las tareas NER y ETD, indicando que para estas tareas los anotadores son precisos pero les faltó identificar algunas entidades y disparadores de eventos.

Según la métrica F1 (tabla 3.7), las tareas en las que se obtuvo mayor acuerdo son NER (73.5), VE (90.5) y ETD (77.5), todas ellas tareas de identificación y clasificación de spans. Las tareas en las que el acuerdo fue peor son RE (24.6) y EAD (44.4).

Al analizar estos resultados, es importante observar que las salidas de las tareas RE y EAD pueden verse afectadas por errores en cascada provenientes de las tareas NER, VE y ETD. Por ejemplo, si una entidad es identificada incorrectamente, todas las relaciones o argumentos de eventos que involucren esa entidad también serán incorrectas. Esto no implica que menores valores en las tareas NER, VE y ETD necesariamente signifiquen menores valores en las tareas RE y EAD. Por ejemplo, si la entidad incorrecta no participa de ninguna relación o argumento de evento, los resultados de estas tareas no se verán afectados. Sin embargo, el hecho de que el modelo produzca entidades, valores o disparadores de eventos incorrectos puede confundirlo y guiarlo a utilizarlos en relaciones o eventos. Dado que este efecto es difícil de cuantificar sin un análisis profundo de cada tarea independiente de los efectos de otras tareas, se le quita prioridad al mismo en el resto del análisis.

Por eso, estos resultados parecen indicar una diferencia en la dificultad de ambos grupos de tareas. Las tareas que implican la identificación y clasificación de spans parecen ser más sencillas para los anotadores. Aquí, también puede influenciar positivamente el uso del asistente de anotación de la herramienta INCEpTION, que produce sugerencias de spans de interés junto con una predicción de sus clases. Por otro lado, las tareas que implican el enlazado de pares de spans y clasificación del enlace parecen ser las de mayor dificultad. Estas observaciones también están respaldadas por apreciaciones personales de los anotadores participantes sobre la dificultad individual de cada tarea.

El hecho de que el acuerdo haya sido consistentemente menor para todos los anotadores en las tareas RE y EAD indica no una dificultad individual, sino una dificultad de la tarea o potenciales deficiencias en su planteamiento o en la asistencia dada a los

anotadores. Esta observación es útil para trabajar en mejoras en la guía de etiquetado y reforzar el entrenamiento de los anotadores para estas tareas.

También se observa que mientras mayor fue la cantidad de documentos anotados por anotador (ver tabla 3.4), mejor es el acuerdo con el autor. Esto puede reforzar la hipótesis de que mientras más experiencia tiene el anotador, mejor es la calidad de sus anotaciones.

Anotador	**# documentos**	**NER**	**VE**	**RE**	**ETD**	**EAD**
Anot. 1	57	80.6	91.3	50.0	95.5	79.5
Anot. 2	21	74.9	92.7	80.9	85.7	76.0
Anot. 3	16	78.0	87.2	33.7	77.1	8.6
Anot. 4	6	82.6	93.3	14.2	82.2	30.0
Promedio	-	79.0	91.1	44.7	85.1	48.5

Tabla 3.5: Acuerdo entre los anotadores independientes y el autor según la métrica precisión. Para cada una de las tareas, se mide la métrica precisión considerando el autor como el *gold standard*. Un valor de 100 quiere decir que todas las anotaciones del anotador son correctas con respecto al autor.

Anotador	**# documentos**	**NER**	**VE**	**RE**	**ETD**	**EAD**
Anot. 1	57	75.0	94.0	54.8	80.3	71.3
Anot. 2	21	66.9	93.8	4.7	80.0	72.1
Anot. 3	16	64.8	89.2	18.1	71.4	2.1
Anot. 4	6	69.6	86.6	14.2	66.6	25.0
Promedio	-	69.0	90.9	22.9	74.5	42.6

Tabla 3.6: Acuerdo entre los anotadores independientes y el autor según la métrica recuperación. Para cada una de las tareas, se mide la métrica recuperación considerando el autor como el *gold standard*. Un valor de 100 quiere decir que el conjunto de anotaciones del anotador incluye las anotaciones del autor.

Anotador	**# documentos**	**NER**	**VE**	**RE**	**ETD**	**EAD**
Anot. 1	57	77.6	92.5	51.5	87.2	74.5
Anot. 2	21	70.7	92.0	8.9	82.1	72.7
Anot. 3	16	70.4	87.8	23.6	72.3	3.4
Anot. 4	6	75.5	89.8	14.2	68.5	27.0
Promedio	-	73.5	90.5	24.6	77.5	44.4

Tabla 3.7: Acuerdo entre los anotadores independientes y el autor según la métrica F1. Para cada una de las tareas, se mide la métrica F1 considerando el autor como el *gold standard*. Un valor de 100 es un acuerdo absoluto entre anotador y autor.

Evaluación de la calidad según acuerdo entre anotadores

Otra forma de evaluar la homogeneidad y calidad de las anotaciones es medir el acuerdo entre anotadores. En este caso, dado que la mayoría de los anotadores anotaron pocos documentos, es más difícil encontrar documentos en el solapamiento de sus conjuntos de datos. Por eso, se comparan los Anotadores 2 y 3 contra el Anotador 1, ya que los solapamientos entre estos tienen la mayor cantidad de documentos.

En la tabla 3.8 se observan las métricas de acuerdo entre el Anotador 1 y los Anotadores 2 y 3, quienes anotaron 11 y 12 documentos en común con el Anotador 1 respectivamente.

Las tareas con mayor y peor acuerdo son las mismas que evaluando el acuerdo contra el autor (RE y EAD), remarcando la dificultad de las tareas de extracción de relaciones y argumentos de eventos. También, en estos resultados se visualiza que para esas tareas no hay homogeneidad entre los anotadores: Anotador 1 y 2 coinciden más en los argumentos de eventos (mayor acuerdo en la tarea EAD), y Anotador 1 y 3 coinciden mejor en las relaciones (mayor acuerdo en la tarea RE).

Anot. 1 versus	**# documentos**	**NER**	**VE**	**RE**	**ETD**	**EAD**
Anot. 2	11	72.8	78.0	25.3	80.0	64.6
Anot. 3	12	72.2	73.8	40.0	76.6	22.0
Promedio	-	72.5	75.9	32.65	78.3	43.3

Tabla 3.8: Acuerdo entre el Anotador 1 y los Anotadores 2 y 3 según la métrica F1. Para cada una de las tareas, se mide la métrica F1 considerando el Anotador 1 como el *gold standard*. Un valor de 100 es un acuerdo absoluto.

3.4.3. Curado y Conjunto Final

Debido al bajo nivel de acuerdo con el autor para las tareas RE y EAD, se realizó un proceso de curado de aproximadamente 130 documentos en los cuales se corrigieron manualmente las anotaciones y discrepancias, con el fin de obtener un conjunto final homogéneo y de mejor calidad. Se seleccionaron aleatoriamente documentos anotados por los anotadores 1, 2 y 3.

El resto de la sección detalla el conjunto de datos final obtenido luego de la etapa de curado.

Partición del conjunto final

El conjunto final cuenta con 403 documentos anotados. De este conjunto se tomó una muestra aleatoria de 101 documentos (25 %) para evaluación de los sistemas construidos. Este conjunto de datos de evaluación es utilizado durante todos los experimentos detallados en el capítulo 5 para evaluar el sistema y sus distintas componentes.

El resto del conjunto de datos, con un total de 302 documentos, conforma el conjunto de datos de entrenamiento. Este conjunto de datos es luego utilizado por el sistema construido para mejorar las salidas generadas, como se describe en el capítulo 4. Por otro lado, durante la construcción del sistema, se tomó una muestra aleatoria de 20 documentos etiquetados de este conjunto para conformar un conjunto de datos de desarrollo que permitiera iterar las instrucciones diseñadas, como se explica en la sección 4.5.1.

La figura 3.5 ilustra gráficamente el particionamiento de los datos explicado previamente. En resumen, en la construcción del sistema (ver capítulo 4) se utiliza el conjunto de entrenamiento para iterar la instrucción dada al modelo. En esta etapa, se separa un subconjunto de 20 documentos (conjunto de desarrollo, representado por el color amarillo en la imagen) del conjunto de entrenamiento, que se utiliza para evaluar los distintos refinamientos aplicados a la instrucción. También, en esa etapa, se utilizan los restantes documentos (conjunto azul en la imagen) para alimentar la base de datos vectorial utilizada y proveer estas anotaciones como ejemplos en la instrucción. Luego, en la etapa de evaluación del sistema construido (ver capítulo 5), se utiliza el conjunto

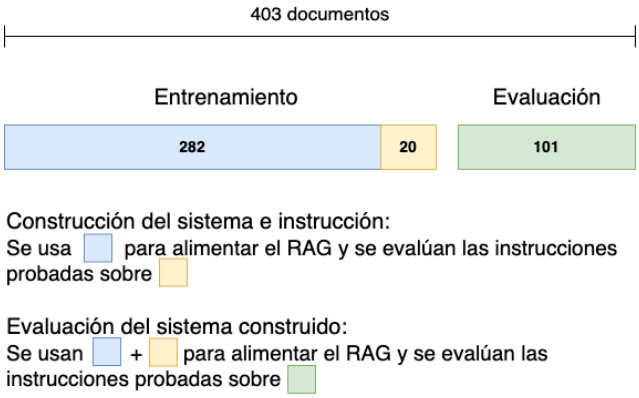


Figura 3.5: Ilustración gráfica del particionamiento de los datos en los conjuntos de entrenamiento, desarrollo y evaluación, junto con una explicación de cuándo y cómo se utiliza cada uno de estos conjuntos.

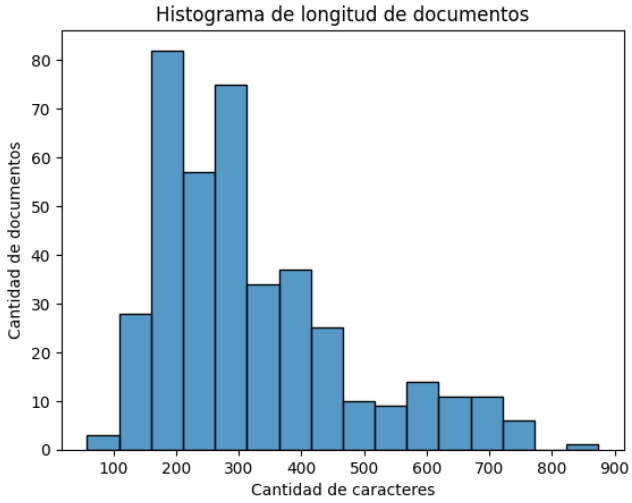


Figura 3.6: Histograma de la cantidad de caracteres por documento.

de entrenamiento completo (conjunto azul más conjunto amarillo) para alimentar la base de datos vectorial, y se evalúa sobre el hasta entonces desconocido conjunto de datos de evaluación (conjunto verde en la imagen).

Estadísticas del conjunto final

A continuación, se presenta un análisis cuantitativo y cualitativo del conjunto construido.

La figura 3.6 muestra un histograma de la cantidad de caracteres por documento. Se observa que la mayoría de los documentos tienen menos de 400 caracteres.

La figura 3.7 muestra la cantidad de anotaciones por tarea y por clase de anotación para el conjunto de datos curado. Debido a la naturaleza de los textos anotados, el conjunto de anotaciones es desbalanceado.

En cuanto a las entidades, existen casi el doble de anotaciones de tipo `Persona` y `Organización` que del tipo `Lugar`. Sin embargo, se cuenta con un número considerable de anotaciones de los tres tipos para realizar evaluaciones significativas o la eventual necesidad de entrenar o realizar fine-tuning de un modelo.

En cuanto a los valores, es mucho más fuerte la presencia de valores de tipo `Tiempo` que de tipo `Causa`. Esto es normal, dado que las fechas ocurren con mucha frecuencia en los textos. Es más, cada ficha debería tener su fecha, lo cual debería contabilizarse como un valor de tipo Tiempo para cada ficha.

La escasez de anotaciones de tipo `Causa` puede resultar problemática, debido a que estos pueden ser valores difíciles de identificar, solo identificables en su contexto, y únicos de este conjunto de datos. En otras palabras, nombres de personas, organizaciones, lugares y fechas están presentes en un sinfín de textos y conjuntos anotados, por lo cual puede ser más sencillo para los LLM extraer estas clases de información. Sin embargo, el tipo `Causa` es propio de este conjunto de datos, y contar con un volumen mayor de anotaciones podría resultar útil para optimizar los modelos o sistemas utilizados para identificarlo mejor.

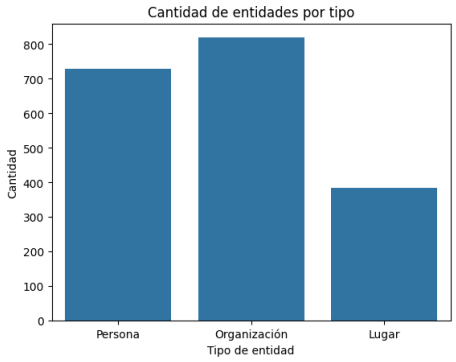
Lo mismo se podría decir de las relaciones. Naturalmente, la relación más frecuente es la relación `integra`, que suele ser utilizada en estas fichas para mencionar la participación de individuos en ciertas organizaciones. Para las demás relaciones (`domicilio`, `contacto` y `familiar`) las cantidades de anotaciones son bajas, lo cual puede provocar que las evaluaciones para estas clases no tengan una significancia estadística o que el funcionamiento para estas clases no pueda ser optimizado.

Para los eventos se cuenta con un poco más de anotaciones. En particular, el evento `Detención` es el más frecuente. Si se observan los argumentos de los eventos, sujeto y fecha suelen ser los más presentes, lo cual es importante para los usos finales de esta información, como poder identificar los participantes de ciertos eventos y en qué momentos ocurrieron. En el evento `Traslado`, el argumento destino también tiene bastante presencia, lo cual podría ser útil para identificar los movimientos de personas en la época (por ejemplo, emigraciones por exilio político). En el evento `Detención`, argumentos como causa, autoridad y destino también tienen presencia, lo cual es útil para obtener mayor información sobre este evento de gran importancia. Lamentablemente, el argumento origen, que también podría utilizarse para reconstruir los recorridos físicos de las personas en la época, no está bien representado en ninguno de los tres eventos.

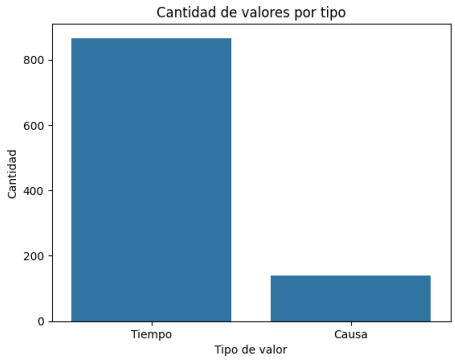
3.5. Reflexiones sobre el proceso de etiquetado y construcción del conjunto de datos

Se alcanzó el principal objetivo de la tarea de etiquetado, que consistía en conseguir un conjunto anotado de documentos de buena calidad para la construcción y evaluación de un sistema de extracción de información. De todas formas, otro gran valor agregado fue obtener retroalimentación del proceso y de la tarea, lo cual permite plantear futuros refinamientos al proceso, las herramientas y la definición de la tarea en sí.

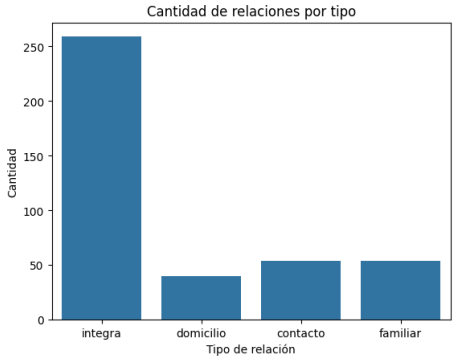
El costo de instruir a un anotador en la tarea de etiquetado y la herramienta hacen que se requiera de un mínimo de anotaciones para que dicho esfuerzo sea fructuoso. Según reflexiones del autor y los anotadores sobre la evolución del proceso de etiquetado, se cree que se requiere de un etiquetado de al menos 50 documentos para obtener la práctica y experiencia necesarias para anotarlos con mayor confianza y sin depender de los materiales de asistencia. Por ende, en posibles iteraciones puede ser útil definir una cuota mínima de documentos por anotador, con el fin de alcanzar las cantidades planteadas, balancear el trabajo y lograr que alcancen la práctica necesaria.



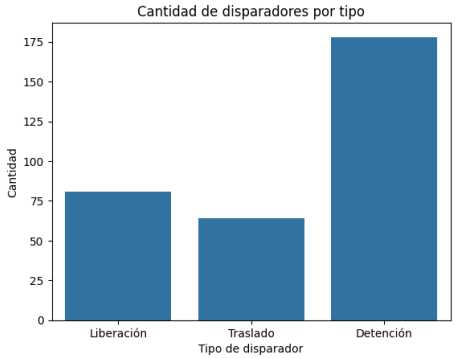
(a) # de entidades



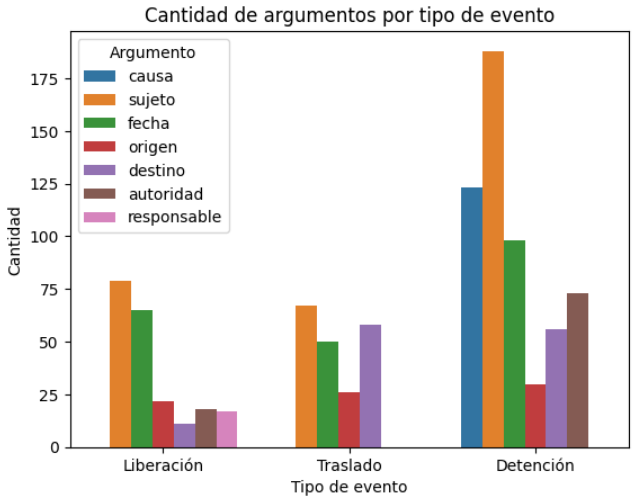
(b) # de valores



(c) # de relaciones



(d) # de disparadores



(e) # de argumentos por tipo de evento

Figura 3.7: Cantidad de anotaciones por tarea y por clase de anotación.

La herramienta INCEpTION demostró ser limitada en varios sentidos. Por ejemplo, no se permite establecer reglas o validaciones a aplicar al etiquetado, o estas funcionalidades son poco flexibles y funcionales. Otras limitantes hacen creer que puede ser de utilidad explorar otras herramientas que ofrezcan mayor flexibilidad y funcionalidades.

Se observó que el etiquetado de relaciones y argumentos de eventos son las tareas más difíciles. Puede ser de utilidad reforzar las secciones de la guía de etiquetado que explican estas tareas y las sesiones de preparación para educar mejor a los anotadores en estas tareas.

Otra forma de mejorar la calidad general del etiquetado y el acuerdo entre anotadores es convertir el proceso en uno iterativo. Realizar una iteración de anotado, calcular las métricas de acuerdo entre anotadores y con el autor, y en base a la retroalimentación obtenida de las métricas, reforzar los puntos débiles de la anotación.

Capítulo 4

Framework para Extracción de Información con Modelos Generativos

El relevamiento del estado del arte revela que la mayoría de los enfoques explorados para la extracción de información con modelos generativos siguen una estructura general de pipeline, implementando cambios menores en distintos componentes. Las implementaciones existentes de estos sistemas no suelen seguir una arquitectura modular, con componentes intercambiables para facilitar la configuración y exploración de distintos pipelines.

En este capítulo se describe el framework desarrollado para la extracción de información utilizando modelos generativos. Una de las características de este framework es su arquitectura modular, que facilita el desarrollo de nuevos componentes y la experimentación de distintas alternativas. A su vez, permite realizar modificaciones a la ontología (por ejemplo, agregar nuevos eventos) sin la necesidad de modificar manualmente la instrucción dada al LLM en diversos puntos. Se comienza por una descripción general del sistema y luego se ahonda en cada una de sus componentes y las variantes implementadas de cada componente.

4.1. Descripción General

El diagrama de la figura 4.1 muestra una vista general del sistema y sus distintos módulos.

El sistema sigue una estructura de pipeline que recibe como entrada el texto plano de una ficha y una descripción estructurada de los esquemas de la información que se desea extraer, con descripciones para cada entidad, relación, evento y sus argumentos. El ejemplo 4.1 muestra el esquema de la relación `Integrates`, en el cual se especifican los componentes de la relación, su significado semántico y sus tipos. Estos esquemas se definen utilizando la librería Pydantic [1] que permite definir modelos de datos y realizar validaciones de datos. Esta librería es ampliamente adoptada por las herramientas existentes para la construcción de soluciones de inteligencia artificial generativa. En particular, Pydantic está fuertemente soportada por la librería LangChain [2] para de-

[1] `https://github.com/pydantic/pydantic`
[2] `https://www.langchain.com/`

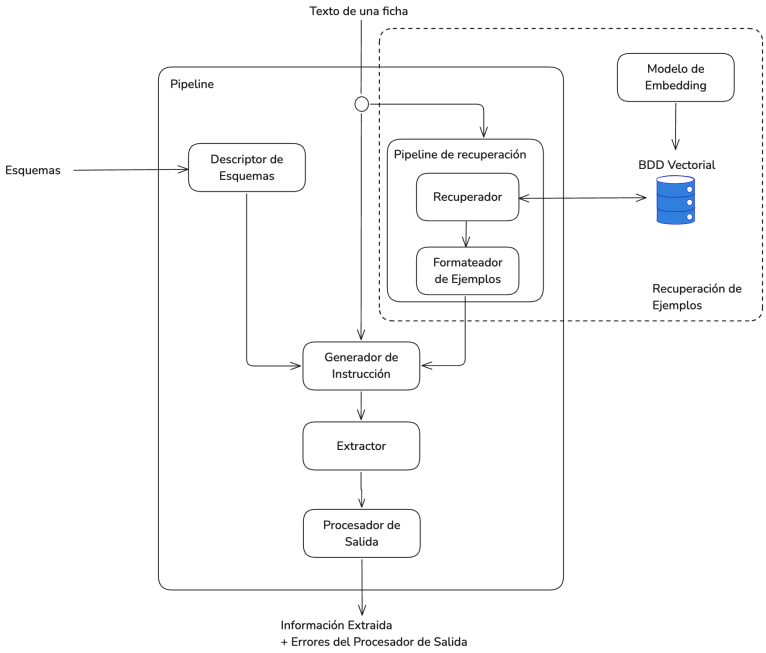


Figura 4.1: Diagrama de la arquitectura del pipeline para la extracción de información.

finir objetos de datos que se espera que generen los LLMs. En este trabajo, se decidió utilizar la librería LangChain dada su popularidad para el desarrollo de soluciones de inteligencia artificial generativa, su flexibilidad y el hecho de que la misma sea *open-source*.

Ejemplo 4.1: Ejemplo de esquema definido en Pydantic para la relación `Integrates`, con descripciones para sus campos.

```
class Integrates(Relation):
    """
    Especifica que una Persona es parte de o trabaja
    para una Organizacion.
    """
    gov: Person = Field(
        id="gov",
        description="La Persona que es parte de o trabaja para
        la Organizacion."
    )
    dep: Organization = Field(
        id="dep",
        description="La Organizacion que integra la Persona."
    )
```

Estos esquemas son definidos para todas las entidades, valores, relaciones y eventos que se desean extraer. Luego, el módulo Descriptor de Esquemas del pipeline se encarga de generar una descripción de texto completa y detallada de estos esquemas que puede ser entregada como entrada a un LLM. Esta descripción, junto con el texto de entrada, se entrega al módulo Generador de Instrucción que combina ambas entradas, junto

con instrucciones relevantes para la tarea, y produce una única instrucción. El módulo Extractor toma la instrucción y se la pasa al LLM, indicándole la tarea a realizar, los esquemas que debe cumplir la información extraída e instrucciones de cómo generar la salida para lograr su correcto procesamiento y generación de instancias de los esquemas provistos.

La salida del Extractor es una cadena de texto que, idealmente, debería seguir el formato de salida especificado en las instrucciones provistas al modelo. Para convertir esta salida de texto a un objeto utilizable programáticamente, se procesa la salida de texto y se instancian los esquemas de datos correspondientes utilizando el módulo Procesador de Salida. En este procesamiento, se descarta cualquier texto generado que no siga las instrucciones que fueron provistas o no cumpla con el formato establecido. El texto que sí cumple con el formato es procesado y utilizado para crear instancias de las clases definidas en los esquemas, que luego son entregadas como salida del sistema.

A su vez, el sistema cuenta con un módulo opcional de Recuperación de Ejemplos que aplica la técnica ICL. Para esto, previamente se generó una base de datos vectorial con los embeddings de los textos de las fichas en el conjunto de datos de entrenamiento. Luego, esta base de datos vectorial es utilizada para recuperar los ejemplos con una mayor similitud semántica al texto de entrada al sistema utilizando el módulo Recuperador, con el fin de utilizar dichos ejemplos y la salida esperada de cada uno para instruir mejor al modelo. Cada ejemplo y su salida esperada son formateados antes de agregarse a la instrucción, utilizando el módulo Formateador de Ejemplos, para seguir el formato con el cual se le solicita al modelo que genere los datos.

Las siguientes secciones describen cada uno de estos módulos y las instancias desarrolladas de cada módulo.

4.2. Esquemas de la Información a Extraer

En este módulo se definen las entidades, los valores, las relaciones y los eventos que se espera extraer del texto. Estos se definen como modelos de Pydantic por las razones expresadas previamente.

Definir estos objetos como modelos de Pydantic tiene además la ventaja de que se agregan automáticamente validaciones de tipos que se ejecutan a la hora de instanciar los objetos. Esto es particularmente útil a la hora de procesar la salida del LLM e instanciar los objetos correspondientes. A pesar de que una descripción de estos modelos se pasa al LLM en la instrucción y se le solicita construir instancias correctas de estos modelos, el LLM puede cometer errores y alucinar. Estos chequeos son necesarios para garantizar que el procesamiento de la salida del LLM sea robusto y confiable, y retorne información correctamente estructurada según los esquemas definidos.

Para todos los esquemas se incluyen descripciones en inglés y español, tanto del modelo como de los campos que lo conforman. Estas descripciones son utilizadas por el módulo 4.3, que retorna una descripción textual de la información a extraer, especificando su estructura y los campos.

Con el fin de poder visualizar estos esquemas, en las siguientes subsecciones solo se incluye la descripción de los modelos y campos, y se excluye código auxiliar. A su vez, las descripciones de los esquemas y sus campos son presentadas en español para una mejor lectura, aunque en el sistema final las mismas son presentadas al LLM en inglés (para una explicación de esta decisión, ver la subsección 4.5.1). Una vista completa de la definición de los esquemas con Pydantic se puede ver en esta carpeta del repositorio [3].

[3] `https://gitlab.fing.edu.uy/mh/information_extraction/-/tree/main/information_extraction/schemas`

4.2.1. Entidades y Valores

En primera instancia, se define el esquema `Span` (ver ejemplo 4.2), que representa una cadena contigua de caracteres.

Ejemplo 4.2: Definición del esquema base `Span`.

```python
class Span(BaseModel):
    """
    Un fragmento de texto asociado con una entidad o valor.
    """
    id: int = Field(
        title="id",
        description="Identificador unico para el span.",
    )
    span: str = Field(title="span", description="El texto del span.")
    begin: int = Field(
        title="begin",
        description="El indice del primer caracter del span en el texto
        completo.",
    )
    end: int = Field(
        title="end",
        description="El indice del ultimo caracter del span en el texto
        completo.",
    )
```

Utilizando esta clase base, luego se definen los esquemas `Entity` y `Value`. El ejemplo 4.3 muestra la definición del esquema base `Entity`.

Ejemplo 4.3: Definición del esquema base `Entity`.

```python
class Entity(Span):
    """
    Una entidad extraida del texto, representada por su primera mencion.
    """
    pass
```

Estas clases base permiten definir los esquemas para las entidades `Person`, `Organization`, `Location` y los valores `Cause` y `Time`. El ejemplo 4.4 muestra la definición del esquema `Person`.

Ejemplo 4.4: Ejemplo de definición del esquema `Person`.

```python
class Person(Entity):
    """
    Una persona o grupo de personas unidas por un contexto.
    Las personas pueden ser mencionadas por su nombre, pronombres,
    titulos, alias u otras referencias.
    """
    pass
```

La definición completa de los esquemas para entidades y valores se puede encontrar

en los archivos `schemas/entities.py` [4] y `schemas/values.py` [5] del repositorio.

4.2.2. Relaciones

Para las relaciones, se define el esquema base `Relation` (ver ejemplo 4.5).

Ejemplo 4.5: Definición del esquema base `Relation`.

```python
class Relation(BaseModel):
    """
    Un par ordenado que representa una relacion entre dos entidades.
    """
    gov: Entity = Field(
        title="gov",
        description="La entidad gobernante en la relacion."
    )
    dep: Entity = Field(
        title="dep",
        description="La entidad dependiente en la relacion."
    )
```

Este modelo base se utiliza para definir los esquemas `Integrates`, `IsFamily`, `Contacted`, `LivesIn` y `Correference`. En estos esquemas es importante definir tipos más específicos en los campos `gov` y `dep`, de forma que las validaciones de los esquemas permitan verificar que no se está intentando crear una relación entre las entidades incorrectas. El ejemplo 4.6 muestra la definición de la relación `Integrates`.

Ejemplo 4.6: Definición del esquema `Integrates`.

```python
class Integrates(Relation):
    """
    Especifica que una Persona es parte de o trabaja
    para una Organizacion.
    """
    gov: Person = Field(
        id="gov",
        description="La Persona que es parte de o trabaja para
        la Organizacion."
    )
    dep: Organization = Field(
        id="dep",
        description="La Organizacion que integra la Persona."
    )
```

Se puede observar que los tipos de los campos `gov` y `dep` están restringidos a los esquemas `Person` y `Organization`, lo cual permite validar que solo se creen relaciones de tipo `Integrates` entre una persona y una organización.

La definición completa de los esquemas para relaciones se puede encontrar en el archivo `schemas/relations.py` [6] del repositorio.

[4] https://gitlab.fing.edu.uy/mh/information_extraction/-/blob/main/information_extraction/schemas/entities.py

[5] https://gitlab.fing.edu.uy/mh/information_extraction/-/blob/main/information_extraction/schemas/values.py

[6] https://gitlab.fing.edu.uy/mh/information_extraction/-/blob/main/information_extraction/schemas/relations.py

4.2.3. Eventos

Para los eventos, se define el esquema base `Event` (ver ejemplo 4.7) que contiene el campo `trigger` y, a su vez, los argumentos que son comunes a todos los eventos.

Ejemplo 4.7: Definición del esquema base `Event`.

```
class Event(BaseModel):
    """
    Un evento extraido del texto.
    """
    trigger: Span = Field(
        id="trigger",
        description="Palabra o frase corta que claramente indica
        la ocurrencia del evento en el texto."
    )
    subject: list[Person] | None = Field(
        id="subject",
        default=None,
        description="La Persona o Personas que son sujeto del evento.
        Si no se mencionan sujetos, este campo es None."
    )
    src: Location | None = Field(
        id="src",
        default=None,
        description="El Lugar donde se origino el evento. Si no se
        menciona el Lugar de origen, este campo es None."
    )
    dst: Location | None = Field(
        id="dst",
        default=None,
        description="El Lugar donde el evento termino. Si no se
        Location el Lugar de destino, este campo es None."
    )
    time: Time | None = Field(
        id="time",
        default=None,
        description="El Tiempo en el que el evento ocurrio. Si no
        se meniona el Tiempo de ocurrencia, este campo es None."
    )
```

Se observa que, al igual que en las relaciones, es necesario definir tipos más específicos para cada argumento y así aprovechar la validación de los esquemas realizada por Pydantic.

Este esquema base se utiliza para definir los modelos `Detention`, `Release` y `Transfer`. El ejemplo 4.8 muestra la definición del evento `Detention`.

Ejemplo 4.8: Definición del esquema `Detention`.

```
    class Detention(Event):
    """
    Detencion de una Persona o grupo de Personas por una autoridad.
    """
    subject: list[Person] | None = Field(
        id="subject",
```

```
        default=None,
        description="La Persona o Personas que fueron detenidas.
        Si no se menciona un sujeto, este campo es None."
    )
    src: Location | None = Field(
        id="src",
        default=None,
        description="El Lugar donde ocurrio la detencion. Si no
        se menciona Lugar donde ocurrio, este campo es None."
    )
    dst: Location | None = Field(
        id="dst",
        default=None,
        description="El Lugar a donde fueron llevados los sujetos
        luego de ser detenidos. Si no se menciona el Lugar al que
        fueron llevados, este campo es None."
    )
    cause: list[Cause] | None = Field(
        id="cause",
        default=None,
        description="La lista de causas de la detencion. Si no
        se mencionan causas, este campo es None."
    )
    authority: Organization | Person | None = Field(
        id="authority",
        default=None,
        description="La Organizacion o Persona que realizo la
        detencion de los sujetos. Si no se menciona una autoridad,
        este campo es None."
    )
```

Se observa que algunos campos se definen nuevamente en este esquema, con el fin de agregar descripciones especificas a este evento de los argumentos comunes a todos los eventos.

La definición completa de los esquemas para eventos se puede encontrar en el archivo `schemas/events.py` [7] del repositorio.

4.2.4. Data

Por último, el esquema `Data` (ver ejemplo 4.9) sirve para encapsular toda la información estructurada contenida dentro de una pieza de texto. Es el objeto de salida del pipeline de extracción de información, y el utilizado a lo largo del proyecto para representar los datos etiquetados.

Ejemplo 4.9: Definición del esquema `Data`.

```
class Data(BaseModel):
    """
    Estructura de datos que permite almacenar toda la informacion
    extraida de un texto.
    """
```

[7] https://gitlab.fing.edu.uy/mh/information_extraction/-/blob/main/information_extraction/schemas/events.py

```
    text: str
    file_id: str
    entities: dict
    values: dict
    relations: list[Relation]
    events: list[Event]
```

4.3. Descriptor de Esquemas

El LLM recibe como entrada una cadena de texto llamada instrucción en la cual se le indica la tarea a realizar y sus restricciones. Para que el LLM entienda qué tipo de información debe extraer, cuándo debe extraerla, y qué restricciones de tipos existen entre las distintas instancias de esta información, se debe proveer en la instrucción una descripción detallada de la información a extraer.

El objetivo del módulo Descriptor de Esquemas es consumir los esquemas descriptos en la sección anterior y generar una cadena de texto que describa los esquemas junto con sus campos, con el fin de incluir dicha descripción dentro de la instrucción que se dará al LLM, describiendo la información a extraer, la estructura de la misma y las restricciones que debe tener en cuenta a la hora de generar la información.

Trabajos previos han evaluado distintas formas de instruir al LLM sobre la información a extraer. Las siguientes subsecciones detallan las instancias desarrolladas de este módulo, siguiendo las líneas de trabajos previos.

4.3.1. Descriptor de Esquemas en Formato Código

Diversos trabajos previos obtuvieron mejores resultados al entregarle al LLM una descripción en código de la información a extraer (ver sección 2.2.2). El argumento detrás de esta idea es que el código es estructurado en naturaleza, y que muchos LLMs están entrenados sobre código y son capaces de generarlo, lo cual facilita que los mismos generen información estructurada cuando la misma se describe y estructura con algún lenguaje de programación.

Siguiendo esta línea, se desarrolló el módulo Descriptor de Esquemas en Formato Código, el cual toma los esquemas, aplica una serie de procesamientos y pasos de limpieza y retorna una cadena de texto que incluye una versión simplificada del código utilizado para definir los esquemas.

Cada uno de estos esquemas contiene una descripción de cuando se debe instanciar dicho esquema. A su vez, cada atributo del esquema contiene su propia descripción. Por ejemplo, para los esquemas de eventos, donde cada argumento de evento es un atributo, la descripción de estos atributos explica el significado de ese argumento y cuando se debe instanciar.

En el anexo B se puede observar un ejemplo completo de la instrucción dada al LLM que utiliza este descriptor.

4.3.2. Descriptor de Esquemas en Formato Lenguaje Natural

Muchos de los LLMs populares están optimizados para trabajar con y generar texto en lenguaje natural. Por esta razón, se exploró la alternativa de instruir al LLM a generar oraciones en lenguaje natural con un formato fijo con el fin de extraer la información del texto. Si bien estas oraciones tienen estructura, dicha estructura está codificada en lenguaje natural y es relativamente sencilla de traducir a los modelos de datos definidos mediante expresiones regulares.

Para el caso de las entidades y valores, se provee en la instrucción el nombre y descripción de la entidad o valor. A su vez, para instanciar una entidad o valor, el LLM debe generar una oración con el formato:

`<ENTITY> is a <ENTITY_TYPE>`

Donde `<ENTITY>` es el span de texto de la mención a la entidad y `<ENTITY_TYPE>` el tipo de la misma. El ejemplo 4.10 muestra la sección de la instrucción dedicada a describir el tipo de entidad `Person`

Ejemplo 4.10: Sección de la instrucción de lenguaje natural dedicada a describir el tipo de entidad `Persona` y su statement asociado.

```
- Person: A person or group of people joined by some context.
People can be mentioned by their name, pronouns, titles,
aliases or other references. A statement that identifies an
entity of this class must have the following format:
<ENTITY> is a Person.
```

Para el caso de las relaciones la sección es similar. Se provee en la instrucción el nombre y descripción de la relación junto con los tipos de entidad que conecta dicha relación. A su vez, se instruye al LLM a generar oraciones con el siguiente formato:

`<ENTITY_A_TYPE> <ENTITY_A> <RELATION_STRING> <ENTITY_B_TYPE> <ENTITY_B>`

Donde `<ENTITY_A>` y `<ENTITY_A_TYPE>` son el span de texto de la mención a la entidad gobernante y su tipo, `<ENTITY_B>` y `<ENTITY_B_TYPE>` son el span de texto de la mención a la entidad dependiente y su tipo, y `<RELATION_STRING>` es un texto corto en lenguaje natural que representa a la relación. El ejemplo 4.11 muestra la sección de la instrucción dedicada a describir el tipo de relación `IsFamily`.

Ejemplo 4.11: Sección de la instrucción de lenguaje natural dedicada a describir el tipo de relación `IsFamily` y su statement asociado.

```
- IsFamily: Specifies that two Persons are family-related.
This applies to direct family, extended family or inlaws.
This relation connects an entity of the class Person to an
entity of the class Person. A statement that identifies a
relation of this class must have the following format:
Person <ENTITY_A> is family of Person <ENTITY_B>
```

Para el caso de los eventos la sección es más compleja. Se provee el nombre y descripción del evento, y a su vez el nombre de cada argumento junto con el tipo de datos del argumento y una descripción. También, el formato de oración utilizado para instanciar el evento depende del tipo de evento. El ejemplo 4.12 muestra la sección de la instrucción dedicada a describir el tipo de evento `Detention`.

Ejemplo 4.12: Sección de la instrucción de lenguaje natural dedicada a describir el tipo de evento `Detention` y su statement asociado.

```
- Detention: Detention of a Person or group of Person
entities by an authority. This event has the following
arguments:
* trigger: this field is a string. Word or phrase that
clearly indicates the occurrence of the event on the text.
* Subject: this field is a comma-separated list of Person
entities. The Person or Persons that are detained.
* Authority: this field is an Organization or Person entity.
The Organization or Person that detained the subjects.
* Time: this field is a Time entity. The time at which the
```

```
detention occurred.
* Cause: this field is a comma-separated list of Cause
entities. The reasons for the detention.
* Source: this field is a Location entity. The Location
where the detention occurred.
* Destination: this field is a Location entity. The Location
where the detained subjects were taken.
A statement that identifies an event of this class must have
the following format:
Person <ENTITY> (or a comma-separated list of Person
entities) was (or 'were' if there's more than one person)
detained by Organization or Person <ENTITY> at Time <VALUE>
and Location <ENTITY> because of Cause <VALUE> (or a
comma-separated list of Cause), and was (or 'were')
tranferred to Location <ENTITY>. The trigger is <TRIGGER>.
```

En el anexo B se puede observar un ejemplo completo de la instrucción que utiliza este descriptor.

4.4. Recuperación de Ejemplos

La técnica ICL (ver sección 2.2.2) demostró tener buenos resultados en múltiples trabajos previos donde se aplican estrategias few-shot [17]. Esta técnica consiste en agregar dentro de la instrucción ejemplos de los datos de entrenamiento junto con sus salidas esperadas, para instruir al LLM con ejemplos de su funcionamiento esperado. Esto permite que el LLM entienda mejor la tarea que se le está solicitando y el formato en el que debe generar su salida.

Inspirado por trabajos recientes [4, 21, 36, 44, 92], se decidió agregar un módulo opcional llamado Recuperación de Ejemplos que mezcla la técnica ICL con RAG para hacer una mejor selección de los ejemplos a incluir en la instrucción. Para esto, se construyó una base de datos vectorial a partir de los embeddings de los textos de los datos de entrenamiento. A la hora de extraer información de un nuevo texto en el pipeline, se obtiene su embedding y se buscan textos con un embedding similar en la base de datos vectorial. Los textos obtenidos, junto con su salida esperada, son posprocesados y agregados a la instrucción. El objetivo es proveer al LLM con ejemplos de cómo se resuelve el problema de extracción de información para textos similares al nuevo texto de entrada.

En las siguientes subsecciones se describen los distintos componentes que forman parte del módulo Recuperación de Ejemplos y que implementan las técnicas de ICL y RAG.

Modelo de Embeddings

Este modelo toma el texto y lo transforma en una representación vectorial de largo fijo que codifica el contenido semántico y sintáctico del texto. El modelo es utilizado para obtener representaciones vectoriales de todos los textos de los datos de entrenamiento, y para obtener el embedding del texto de entrada a la cadena.

Durante la experimentación se decidió utilizar el modelo de embeddings BGE-M3 [12] por ser de libre acceso, eficiente y poder embeber textos en español y de diversos largos. A su vez, experiencias previas del autor con el modelo facilitaron su rápida integración dentro del sistema. Un breve análisis sobre la calidad de los embeddings de los datos de entrenamiento se puede encontrar en el anexo C.

El modelo de embeddings es fácilmente intercambiable y configurable dentro del pipeline. La experimentación de distintos modelos de embeddings no fue parte del alcance de este trabajo. Sin embargo, idealmente se debería experimentar con múltiples modelos para encontrar el que obtenga los mejores resultados para el conjunto de datos utilizado.

Base de datos vectorial

En todos los experimentos se utilizó la base de datos vectorial Milvus [74] por su facilidad de configuración. Este módulo también es fácilmente intercambiable, habilitando la posibilidad de experimentar con otras bases de datos vectoriales que ofrezcan distintos métodos de búsqueda de vectores o mayor eficiencia.

Recuperador

Este módulo se encarga de obtener el embedding del texto de entrada y realizar la búsqueda por similitud de ejemplos en la base de datos vectorial. En este módulo se pueden utilizar los distintos métodos de búsqueda por similitud disponibles en la librería LangChain.

En este trabajo, solo se experimentó con el algoritmo de búsqueda por similitud básico, que compara el embedding del texto de entrada y los embeddings de los ejemplos de la base de datos vectorial y obtiene los k ejemplos con puntaje de similitud más alto. Se experimentó con distintos valores de k y sus efectos en los resultados de salida del pipeline para los datos del conjunto de evaluación. Los resultados de estos experimentos se detallan en la sección 5.2.

Datos utilizados

Esta técnica requiere de datos etiquetados que son seleccionados por el módulo Recuperador e incluidos dentro de la instrucción que luego será dada al LLM.

Durante la construcción y desarrollo del sistema, se alimenta al módulo de recuperación de ejemplos con un subconjunto de los datos de entrenamiento, con el fin de evaluar distintas variantes del sistema y la instrucción utilizada sobre un conjunto de desarrollo extraído del conjunto de entrenamiento (ver sección 3.4.3 para una explicación del particionamiento de datos, y la sección 4.5.1 para una descripción del proceso de construcción de la instrucción).

Durante la evaluación del sistema (ver capítulo 5), se alimenta este módulo con el conjunto completo de datos de entrenamiento.

4.4.1. Formateador de Ejemplos

Una vez seleccionados los ejemplos a incluir en la instrucción, el texto de los mismos y la salida esperada se deben procesar para seguir el formato esperado de salida del modelo. Esto ayuda a instruir al LLM no solo sobre la información que debe extraer para cada ejemplo, sino sobre cómo debe generar la salida para luego procesarla en el módulo Procesador de Salida.

El formato de los ejemplos está fuertemente atado al tipo de instrucción utilizado y a la descripción de los esquemas: si se espera que el LLM genere texto que represente la información extraída, la salida esperada de los ejemplos debería seguir el mismo formato.

A continuación se detallan las distintas instancias desarrolladas del módulo Formateador de Ejemplos.

Formateador de Ejemplos en Formato Código

Esta componente se utiliza en conjunto con el descriptor de esquemas Descriptor de Esquemas en Formato Código e instancia la información a extraer como objetos de Python. Cada pieza de información a extraer se representa como una nueva instancia de objeto que se asigna a una variable. En el ejemplo 4.13 se observa el texto y salida esperada de una ficha de los datos de entrenamiento formateados con este módulo.

Ejemplo 4.13: Ejemplo procesado por el módulo Formateo de Ejemplos en Formato Código.

````
Here are some examples:

Example ----------------------------------------------------
```python
# This is the example text
text = """
Causante: Milton Osvaldo BALEIRON

Nro. Ficha: 22092

Fecha: 6—9—73

Texto: ANTECEDENTES: Encuadramiento: "Funciono" en la Col.70,
se encuentra detenido.—

Referencias: None
"""

# These are the output instances for this example
entity1 = Person(span='Milton Osvaldo BALEIRON', other_spans=[])
entity2 = Organization(span='Col.70', other_spans=[])
value1 = Time(span='6—9—73')
relation1 = Integrates(gov=entity1, dep=entity2)
event1 = Detention(trigger='detenido', subject=[entity1])
```
````

En la instrucción, los ejemplos son señalizados por la palabra “Example” seguida de un separador de guiones.

Se puede observar cómo los objetos que pertenecen a modelos de relaciones o eventos referencian a objetos de entidades o valores instanciados previamente en el código mediante sus nombres de variables. Esto es particularmente útil para darle al LLM la señal de que algunas entidades o valores pueden participar en múltiples relaciones o eventos, y es importante no instanciar múltiples veces la misma entidad o valor, sino que se deben usar referencias a la ya existente.

Este ejemplo de salida también permite instruir al LLM sobre el formato esperado. Luego, el módulo de Procesador de Salida se encarga de leer dicha salida e instanciar los modelos de datos correspondientes, con el fin de recuperar la información estructurada a partir de la salida potencialmente no estructurada del LLM.

Formateador de Ejemplos en Formato Lenguaje Natural

Esta componente se utiliza en conjunto con el descriptor de esquemas Descriptor de Esquemas en Formato Lenguaje Natural, e instancia la información a extraer con oraciones que siguen el formato especificado en la sección 4.3.2. En el ejemplo 4.14

se observa el texto y salida esperada de una ficha de los datos de entrenamiento formateados con este módulo.

Ejemplo 4.14: Ejemplo procesado por el módulo Formateo de Ejemplos en Formato Lenguaje Natural.

```
Example ------------------------------------------------
-> This is the example text:
Causante: TERESA ANGELICA BRIOZZO CLIVIO

Nro. Ficha: 470

Fecha: 22-8-71

Texto: ANTECEDENTES: Detenida por el G.A.1 el 16/Agosto/973
en su domicilio, en averiguaciones por Presuncion de
participar en el M.L.N. (Tupamaros).- Alojada en el G.A.1.-
Ver documentos (No. 360) del G.A.1 del 16 de agosto de 1973

Referencias: None

-> These are the statements:
List of entities:
TERESA ANGELICA BRIOZZO CLIVIO is a Person
G.A.1 is an Organization
su domicilio is a Location
M.L.N. (Tupamaros) is an Organization
G.A.1. is a Location

List of values:
22-8-71 is a Time
16/Agosto/973 is a Time
Presuncion de participar en el M.L.N. (Tupamaros) is a Cause
16 de agosto de 1973 is a Time

List of relations:
Person TERESA ANGELICA BRIOZZO CLIVIO lives in Location su
domicilio
Person TERESA ANGELICA BRIOZZO CLIVIO integrates
Organization M.L.N. (Tupamaros)

List of events:
Person TERESA ANGELICA BRIOZZO CLIVIO was detained by
Organization G.A.1 at Time 16/Agosto/973 and Location
su domicilio because of Cause Presuncion de participar
en el M.L.N. (Tupamaros), and was transferred to Location
G.A.1.. The trigger is Detenida.

End of example ----------------------------------------
```

En esta instrucción los ejemplos también son señalizados por la palabra “Example”, seguida de un separador de guiones, y terminados por la frase “End of example”, también con un separador.

4.5. Generador de Instrucción

La instrucción a utilizar con el LLM debe integrar tres variables distintas: el texto de entrada para el cual se desea extraer la información, la descripción de los esquemas proveniente del Descriptor de Esquemas y los ejemplos procesados provenientes del módulo de Recuperación de Ejemplos. Esta última variable es opcional y puede no incluirse en casos en que no se desee utilizar la técnica de ICL, lo cual es una variante con la que se experimenta. El módulo Generador de Instrucción se encarga de tomar estas variables y producir una única instrucción que las combina.

Como se mencionó previamente, la instrucción utilizada debe estar alineada con el Descriptor de Esquemas y el módulo de Formateo de Ejemplos seleccionado, ya que todo decanta en una única instrucción de texto que incluye las salidas de estos módulos.

Por lo tanto, se desarrolló el módulo Generador de Instrucción de forma que pueda producir dos tipos de instrucciones:

- Instrucción de Código: esta instrucción le indica al LLM que recibirá una descripción en código de clases que representan la información a extraer y que debe generar instancias de dichas clases, cumpliendo las tareas de extracción de información correspondientes.
- Instrucción en Lenguaje Natural: esta instrucción le indica al LLM que recibirá una descripción textual de la información a extraer y que debe generar oraciones con una estructura fija ("*statements*") que instancian la información a extraer.

Una versión completa y extensa de ambas instrucciones se puede encontrar en el anexo B.

4.5.1. Elaboración de las instrucciones

La construcción de las instrucciones a utilizar en el LLM es de alta importancia para lograr buenos resultados [59]. En esta labor, no basta solo con proveer las instrucciones sobre la tarea al modelo, sino que la claridad, el formato de la escritura, las directivas o palabras utilizadas, los ejemplos dados y la extensión de la instrucción son factores que pueden influir en los resultados obtenidos por el modelo. Esta labor, conocida popularmente como *prompt engineering*, se logra mediante un proceso de prueba, error y evaluación en el cual se modifican levemente estos factores de la instrucción, se evalúa su efectividad y se itera el proceso. Al ser un proceso altamente experimental, sin contar con un conjunto de reglas que permitan mejorar la instrucción de forma determinista, este debe ser acotado en tiempo y esfuerzo al alcance del proyecto.

Ambos tipos de instrucciones utilizadas en este trabajo fueron iteradas y construidas siguiendo este proceso. Para evaluar la efectividad de las instrucciones probadas, se utilizó el conjunto de datos de desarrollo detallado en la sección 3.4.3. Este conjunto de desarrollo se formó tomando una muestra de 20 ejemplos etiquetados seleccionados aleatoriamente para validar cambios y modificaciones grandes en la instrucción. Modificaciones grandes refieren al agregado de secciones completas a la instrucción, directivas particulares que restrinjan o soliciten ciertas acciones, o el uso de distintos idiomas. El resto de los documentos del conjunto de entrenamiento se utilizaron para el módulo de recuperación de ejemplos en estas pruebas.

Un ejemplo concreto de decisión de diseño tomada durante esta etapa de construcción de las instrucciones fue el uso del idioma inglés para la escritura general de la instrucción y la descripción de la ontología. Al experimentar con los idiomas inglés y español, rápidamente se observó una diferencia de calidad significativa en los resultados de los LLM al usar uno u otro idioma. En el caso de español, los modelos parecían

Modelo	Tipo	Parámetros (b)	Tamaño (GB)
CodeGemma [66]	Code-LLM	7	5.0
Qwen2.5-Coder [86]	Code-LLM	14	9.0
Deepseek-Coder-v2 [94]	Code-LLM	16	8.9
Llama3.1 [18]	NL-LLM	8	4.9
Qwen2.5 [86]	NL-LLM	14	9.0
Deepseek-R1 [20]	NL-LLM	14	9.0

Tabla 4.1: Modelos utilizados para la experimentación, junto con su tipo (Code-LLM vs NL-LLM), cantidad de parámetros y tamaño en GBs.

no entender claramente las directivas de la tarea, produciendo salidas completamente erróneas en muchos casos. Por esta razón, en etapas tempranas del desarrollo del trabajo se decidió continuar con un solo idioma, ya que mantener todas las descripciones de la ontología, el formateo de los ejemplos y la estructura general de la instrucción en ambos idiomas resultaría en un esfuerzo elevado, mientras que la instrucción en español no presentaba indicios de producir buenos resultados. Experimentar con modelos más grandes y capaces que puedan comprender mejor la instrucción en español es una línea de trabajo interesante para el futuro.

Como se evidencia en la experimentación llevada a cabo (capítulo 5), el sistema cuenta con un alto número de parámetros o grados de libertad que pueden afectar, en distintas medidas, los resultados finales obtenidos. De todos estos parámetros, se cree que la escritura general de la instrucción no afecta significativamente los resultados, y por cuestiones de alcance del trabajo y cantidad de esfuerzo, no se incluye en dicho capítulo una experimentación rigurosa con múltiples variantes de escritura de las instrucciones. Sin embargo, es pertinente mencionar que las versiones presentes de las instrucciones fueron iteradas y refinadas siguiendo el proceso mencionado anteriormente por un periodo acotado de tiempo durante el desarrollo del sistema.

4.6. Extractor

El módulo Extractor toma la instrucción producida por el Generador de Instrucción y utiliza un LLM para generar una respuesta extrayendo la información especificada en los esquemas.

Debido a restricciones de privacidad de datos, solo se experimenta con modelos open-source que puedan correr de forma local.

Siguiendo la nomenclatura utilizada en otros trabajos del área, se clasifican los LLMs explorados en dos tipos:

- NL-LLMs: Modelos entrenados sobre textos de diversos dominios, especializados en comprender y producir texto en lenguaje natural. También pueden haber sido entrenados sobre código, pero no estan especializados para este tipo de dato.
- Code-LLMs: Modelos que parten como NL-LLMs pero son entrenados sobre grandes corpus de código, especializados en comprender instrucciones para completar tareas de código.

La tabla 4.1 detalla los distintos LLMs probados, los clasifica y especifica sus tamaños.

4.7. Procesador de Salida

La salida del Extractor es texto no estructurado. Este texto puede incluir código y, si el LLM sigue las instrucciones y ejemplos dados en la instrucción, debería contener definiciones de las instancias de la información extraída del texto. De forma de poder utilizar la información extraída en otras aplicaciones o poder almacenarla de forma estructurada, se debe procesar la salida del modelo para instanciar los modelos de datos definidos en los esquemas.

Sin embargo, debido a problemas de alucinación, este texto puede contener texto o código que incumple las instrucciones pedidas, instancias de la información que no cumplen los esquemas definidos o incluso otros errores inesperados. Por ende, se debe construir un módulo que procese la salida del modelo y produzca datos a partir del texto siguiendo los esquemas establecidos, de manera robusta a los posibles errores introducidos por el LLM.

Este módulo toma como entrada la respuesta del módulo de extracción y retorna la información extraída del texto encapsulada en una instancia del modelo de datos `Data`. También, se retorna una lista con todas las excepciones u errores encontrados durante el procesamiento de la respuesta y la instanciación de los modelos de datos.

4.7.1. Procesador de Salida de Código

Este módulo toma como entrada la respuesta del LLM que, siguiendo con las indicaciones de la instrucción de código, debería retornar código de Python que instancie las clases que representan la información extraída y las asigne a variables, siguiendo el formato establecido. El ejemplo 4.15 detalla la sección de la instrucción de código donde se explica el formato que debe seguir la salida del LLM.

Ejemplo 4.15: Sección de la instrucción de código donde se explica el formato que debe seguir la salida del LLM.

```
Your output must have the following format:
* You must instantiate one class per line
* You must assign the instance to a variable with the name
‘entity<index>‘, ‘value<index>‘, ‘event<index>‘, or
‘relation<index>‘, where ‘<index>‘ is a number starting from
1.
Here’s an example of the expected output format:
‘‘‘python
# Entities
entity1 = Person(<ARGUMENTS>)
entity2 = Location(<ARGUMENTS>)
# Values
value1 = Date(<ARGUMENTS>)
value2 = Cause(<ARGUMENTS>)
value3 = Date(<ARGUMENTS>)
# Relations
relation1 = LivesIn(<ARGUMENTS>)
relation2 = Integrates(<ARGUMENTS>)
# Events
event1 = Detention(<ARGUMENTS>)
‘‘‘
```

El ejemplo 4.16 muestra una salida generada correctamente por el LLM para la instrucción de código, cumpliendo con el formato especificado en el ejemplo 4.15.

Ejemplo 4.16: Ejemplo de respuesta generada correctamente por el LLM para la instrucción de código.

```
```python
# Entities
entity1 = Person(span='juan rodriguez', other_spans=[])
entity2 = Organization(span='Col.7.', other_spans=[])
entity3 = Organization(span='Grupo de "Poco"', other_spans
    ↪ =[])
entity4 = Location(span='Zona 2', other_spans=[])
# Values
value1 = Cause(span='"Funciono"')
value2 = Time(span='6-9-73')
# Relations
relation1 = Integrates(gov=entity1, dep=entity2)
relation2 = Integrates(gov=entity1, dep=entity4)
# Events
event1 = Detention(trigger='Encuadramiento', subject=[
    ↪ entity1],
cause=[value1], time=value2)
```
```

La salida del Procesador de Salida de Código en este caso sería un objeto de tipo `Data`, con instancias de estos modelos de datos.

El funcionamiento es similar a ejecutar estas líneas de código en un intérprete y asignar las variables dentro de una instancia de `Data`. Sin embargo, esto no es recomendable, ya que ejecutar código generado por un LLM puede resultar en un enfoque poco robusto y con vulnerabilidades de seguridad. Por ende, se optó por hacer un procesamiento de la respuesta con expresiones regulares para extraer las instancias que se intentan crear y crearlas en un flujo más controlado. Las expresiones regulares utilizadas por el Procesador de Salida de Código se detallan en el anexo E.

Se observa que dicho procesamiento es bastante rígido. Por lo tanto, si el LLM alucina (por ejemplo, inventa modelos de datos que no fueron descriptos en la instrucción), utiliza sinónimos para los nombres de los modelos (por ejemplo, en vez del modelo de datos `Time` le pone el nombre `Date`), o comete errores de instanciación (por ejemplo, asigna un argumento a un evento con un tipo incorrecto), esto resultará en errores durante el procesamiento que eliminan esa instancia del resultado final, pero no detienen el procesamiento del resto de la información extraída. En el ejemplo 4.17 se puede observar una respuesta con errores del LLM, donde se inventa un modelo de datos que no existe y se agrega un argumento a un evento con un tipo incorrecto.

Ejemplo 4.17: Ejemplo de salida del LLM para la instrucción de código, con errores introducidos por el LLM.

```
```python
# Entities
entity1 = Person(span='juan rodriguez', other_spans=[])
# Values
value1 = Time(span='6-9-73')
value2 = Money(span='$35.00')
# Events
event1 = Detention(trigger='Encuadramiento', subject=[
    ↪ entity1], cause=[value1])
```
```

Dada la salida del LLM del ejemplo 4.17, la salida del procesador de salida es una instancia de `Data` que incluye:

- Una entidad:

```
Person(span='juan rodriguez')
```

- Un valor:

```
Time(span='6-9-73')
```

- Un evento:

```
Detention(
    trigger='Encuadramiento',
    subject=[Person(span='juan rodriguez')]
)
```

La salida también incluye una lista de dos errores, indicando que se intentó instanciar un modelo inexistente y que se intentó agregar un argumento de tipo inválido al evento de detención.

4.7.2. Procesador de Salida de Lenguaje Natural

En caso de que se esté utilizando la instrucción de lenguaje natural, la salida esperada del modelo es diferente. En este caso, se espera que el modelo genere *statements* que luego se pueden procesar utilizando expresiones regulares para instanciar los modelos de datos correspondientes. Las expresiones regulares utilizadas por el Procesador de Salida de Lenguaje Natural se detallan en el anexo E.

El ejemplo 4.18 muestra una posible respuesta de un LLM para la instrucción de lenguaje natural.

Ejemplo 4.18: Ejemplo de salida del LLM para la instrucción de lenguaje natural.

```
List of entities:
TERESA ANGELICA BRIOZZO CLIVIO is a Person
G.A.1 is an Organization
M.L.N. (Tupamaros) is an Organization

List of values:
22-8-71 is a Time
16/Agosto/973 is a Time
Presuncion de participar en el M.L.N. (Tupamaros) is a Cause

List of relations:
Person TERESA ANGELICA BRIOZZO CLIVIO integrates
    ↪ Organization M.L.N. (Tupamaros)

List of events:
Person TERESA ANGELICA BRIOZZO CLIVIO was detained by
    ↪ Organization G.A.1 at Time 16/Agosto/973 and <UNKNOWN>
    ↪  because of Cause Presuncion de participar en el M.L.N
    ↪ . (Tupamaros), and was transferred to Location G.A.1..
    ↪  The trigger is Detenida.
```

Al recibir esta respuesta, el procesador validará que siga las reglas y esquemas definidos en la instrucción, y producirá una instancia de `Data` con la información extraída, que incluye, por ejemplo:

- Una entidad:

```
Person(span='TERESA ANGELICA BRIOZZO CLIVIO')
```

- Un valor:

```
Time(span='16/Agosto/973')
```

- Una relación:

```
Integrates(
    gov=Person(span='TERESA ANGELICA BRIOZZO CLIVIO'),
    dep=Organization(span='M.L.N. (Tupamaros)')
)
```

- Un evento:

```
Detention(
    trigger='Detenida',
    subject=[Person(span='TERESA ANGELICA BRIOZZO CLIVIO')],
    authority=Organization(span='G.A.1'),
    time=Time(span='16/Agosto/973'),
    cause=[
        Cause(
            span='''Presuncion de participar
            en el M.L.N. (Tupamaros)'''
        )
    ],
    dst=Location(span='G.A.1.')
)
```

El Procesador de Salida de Lenguaje Natural, al igual que el de código, debe ser robusto a errores o alucinaciones del LLM. Ante statements que no sigan las instrucciones especificadas o referencias erróneas a tipos inexistentes u objetos no declarados, omitirá dichos statements del procesamiento y agregará los errores encontrados a la lista de errores de salida.

4.7.3. Contabilización de errores generados por el LLM

Los errores detectados por el Procesador de Salida son almacenados para un análisis más detallado que se presenta en la sección 5.1.4.

La tarea del Procesador de Salida es instanciar piezas de información (eventos, valores, relaciones, eventos o argumentos de eventos) en base a la salida de texto del LLM. El LLM tiene dos formas de instanciar una pieza de información dependiendo de la instrucción que se utilice:

- En la instrucción de código, el LLM instancia piezas de información al definir el código que produce una instancia de una clase y la asigna a una variable.

- En la instrucción de lenguaje natural, el LLM instancia piezas de información utilizando un statement.

Se considera un error generado por el LLM si, al momento de instanciar una pieza de información, se encuentra un error de alguno de estos tipos:

- De formato: el LLM produce texto que no sigue el formato válido para instanciar la información extraída. En el caso de la instrucción de código, esto puede darse al producir código con una sintaxis incorrecta. En el caso de la instrucción de lenguaje natural, esto puede darse al producir un statement con un formato incorrecto.
- De dominio: el LLM intenta instanciar un tipo de información que no fue definido dentro del dominio de información que debe extraer. Por ejemplo, intentar instanciar una entidad de tipo `Documento`.
- De tipo: el LLM intenta instanciar una relación o un argumento de evento con tipos inválidos. Por ejemplo, intentar crear una relación de tipo `domicilio` entre dos instancias de tipo `Organización`.
- De referencia: el LLM referencia a una instancia de información que no fue instanciada previamente o que fue instanciada incorrectamente. Por ejemplo, si una relación refiere a una entidad que no fue definida previamente o que fue definida incorrectamente por un error de formato.

Al detectar un error de este tipo para una instancia de información, dicha instancia se ignora y se descarta de la salida final del sistema.

Esto implica que entidades o valores definidos incorrectamente pueden provocar errores en cascada. Por ejemplo, si el LLM produce un statement con un formato incorrecto para una entidad, dicha entidad es ignorada de la salida final. Consecuentemente, todas las relaciones o argumentos de eventos que involucren la entidad incorrectamente definida también se encuentran mal definidos y son ignorados de la salida final. En este caso, el error en la definición de la entidad o valor se contabiliza múltiples veces: una vez por la definición incorrecta de la entidad o valor, y otra vez por cada una de las relaciones o argumentos de los que esa entidad o valor participa.

Los argumentos de los eventos se evalúan independientemente de los demás argumentos del mismo evento. Esto quiere decir que, si se tiene un argumento de evento incorrectamente definido, se lo elimina de la salida final sin la necesidad de eliminar el evento en su totalidad o de eliminar otros argumentos que hayan estado correctamente definidos. Esto permite no descartar la salida completa del LLM para un evento si uno de sus argumentos estuvo mal definido.

El ejemplo 4.19 muestra una salida generada por el LLM para la instrucción natural que cuenta con múltiples errores:

- El statement “The mention M.L.N. (Tupamaros) is an Organization” sigue un formato incorrecto, y por ende dicha entidad es descartada.
- El statement “P.N.D 12/978 is a Document” refiere a un tipo de valor que no fue definido en el dominio de la información a extraer, y por ende dicho valor es descartado.
- El statement “Person TERESA ANGELICA BRIOZZO CLIVIO integrates Organization M.L.N. (Tupamaros)” refiere a una entidad (“Organization M.L.N. (Tupamaros)”) que fue incorrectamente definida previamente y descartada, y por lo tanto esta referencia también es incorrecta. La relación es descartada de la salida final.

Por lo tanto, la salida del ejemplo 4.19 cuenta con 3 errores.

Ejemplo 4.19: Ejemplo de salida con errores para la instrucción de lenguaje natural.

```
List of entities:
TERESA ANGELICA BRIOZZO CLIVIO is a Person
G.A.1 is an Organization
The mention M.L.N. (Tupamaros) is an Organization

List of values:
22-8-71 is a Time
16/Agosto/973 is a Time
P.N.D 12/978 is a Document

List of relations:
Person TERESA ANGELICA BRIOZZO CLIVIO integrates
    ↪ Organization M.L.N. (Tupamaros)

List of events:
```

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Experimentación

En este capítulo se detallan los experimentos llevados a cabo y se analizan los resultados obtenidos.

Es importante destacar que los datos de evaluación, al igual que los utilizados para construir el sistema, se ven afectados por ruido proveniente de distintas fuentes:

- Errores ortográficos o gramaticales de las fichas originales.
- Errores del proceso de transcripción, debido a la falta de claridad en las imágenes escaneadas o a carencias del proceso de transcripción en sí.

Sumado también a que el lenguaje utilizado en las fichas proviene de otra época y está acotado a un dominio particular, lo cual puede dificultar su comprensión por modelos de lenguaje entrenados sobre español actual y de dominios generales.

A pesar de que se llevó a cabo un proceso de selección y curado de los datos en la etapa de construcción del conjunto de datos, estas fuentes de ruido pueden perjudicar los resultados obtenidos y dificultar la comparación directa con otros conjuntos de datos y sistemas.

Todos los experimentos realizados en este capítulo fueron evaluados sobre el conjunto de datos de evaluación (101 documentos) detallado en la sección 3.4.3. El conjunto de datos de entrenamiento (302 documentos) se almacena en la base de datos vectorial y se utiliza para alimentar el módulo de Recuperación de Ejemplos, como se explica en la sección 4.4.

5.1. Comparación entre distintos modelos e instrucciones

La primera evaluación consiste en comparar el desempeño de múltiples modelos open-source liberados recientemente. La lista completa de modelos se puede ver en la tabla 4.1. A su vez, se evalúan ambos tipos de instrucciones (de código y lenguaje natural) en ambos tipos de modelo (Code-LLM y NL-LLM). Para todas las ejecuciones, se fija la cantidad de ejemplos de ICL en $k = 3$, al ser esta una cantidad de ejemplos que obtiene buenos resultados en promedio sobre el conjunto de evaluación, como se detalla en la sección 5.2.

La temperatura de los LLM se fija en $t = 0{,}1$. Este es un valor relativamente bajo de temperatura que hace que la salida sea más determinista y favorezca a los tokens más probables de la distribución de probabilidad entrenada por el LLM. Esto provoca que el LLM tienda a producir salidas más repetitivas, enfocadas en la tarea solicitada y menos diversas [57]. Por estas razones se eligió un valor bajo de temperatura, aunque

Instrucción	Modelo	Exacta	Fuzzy
Código	CodeGemma	55.0	60.3
	DeepSeek-Coder-v2	52.3	56.6
	Qwen2.5-Coder	**63.0**	**68.8**
	DeepSeek-R1	35.8	39.1
	Llama3.1	42.7	46.6
	Qwen2.5	52.3	56.8
Lenguaje Natural	CodeGemma	44.2	51.0
	DeepSeek-Coder-v2	41.9	47.9
	Qwen2.5-Coder	60.4	65.8
	DeepSeek-R1	54.7	62.3
	Llama3.1	59.4	63.5
	Qwen2.5	60.9	67.0

Tabla 5.1: Resultados de la evaluación para la tarea NER utilizando los dos tipos de instrucción en conjunto con los distintos modelos. Se muestra el promedio ponderado de la métrica F1 utilizando las dos estrategias de emparejamiento.

en futuros trabajos sería interesante evaluar el efecto de utilizar distintos valores de temperatura sobre las salidas finales generadas por el sistema.

Los resultados completos de la evaluación para las tareas *Named-Entity Recognition* (NER), *Value Extraction* (VE), *Relation Extraction* (RE), *Event Trigger Detection* (ETD) y *Event Argument Detection* (EAD) se encuentran en las tablas 5.1, 5.2, 5.3, 5.4, 5.5 respectivamente. Los valores mostrados en estas tablas son el promedio ponderado de la métrica F1 sobre el conjunto de evaluación, explicado previamente en la sección 2.5.7, utilizando las dos estrategias de emparejamiento de spans.

La tabla 5.6 muestra una vista pivote de los resultados que permite comparar de forma más sencilla cuál fue la instrucción que tuvo mejor efecto para cada modelo.

La tabla 5.7 permite analizar la cantidad de errores encontrados por el Procesador de Salida al intentar procesar la salida del LLM. Estos errores pueden deberse a alucinaciones, falta de alineamiento con las instrucciones otorgadas, generación de texto innecesario y otros.

5.1.1. Observaciones sobre las métricas

Para todas las tareas, como es esperable, se observa que la estrategia de emparejamiento Fuzzy obtiene mejores resultados que la estrategia Exacta, dado que la primera es menos restrictiva y permite diferencias de algunos caracteres entre el dato etiquetado y el dato extraído.

Es interesante notar que la ganancia en los valores de las métricas al pasar de la estrategia Exacta a la Fuzzy no es igual para todos los modelos e instrucciones. Esto es particularmente interesante en el caso de la tarea VE (ver tabla 5.2), ya que la mejor instrucción para la tarea cambia dependiendo de qué estrategia de emparejamiento se utilice. Si se utiliza la estrategia de emparejamiento Exacta, la mejor instrucción es la de código; si se utiliza la estrategia Fuzzy, la mejor instrucción es la de lenguaje natural.

Dado que la estrategia Fuzzy solo permite diferencias de algunos caracteres entre la información extraída y la información etiquetada, para el resto de los análisis de esta sección se prefiere la estrategia de emparejamiento Fuzzy, ya que es menos restrictiva y permite que se acepten como válidas pequeñas variaciones de la información extraída

Instrucción	Modelo	Exacta	Fuzzy
Código	CodeGemma	67.1	68.0
	DeepSeek-Coder-v2	61.9	62.9
	Qwen2.5-Coder	**77.1**	78.3
	DeepSeek-R1	39.2	39.8
	Llama3.1	58.3	60.5
	Qwen2.5	74.8	77.1
Lenguaje Natural	CodeGemma	59.7	62.4
	DeepSeek-Coder-v2	64.3	66.0
	Qwen2.5-Coder	76.1	**78.5**
	DeepSeek-R1	63.7	69.8
	Llama3.1	69.2	72.1
	Qwen2.5	70.7	75.5

Tabla 5.2: Resultados de la evaluación para la tarea VE utilizando los dos tipos de instrucción en conjunto con los distintos modelos. Se muestra el promedio ponderado de la métrica F1 utilizando las dos estrategias de emparejamiento.

Instrucción	Modelo	Exacta	Fuzzy
Código	CodeGemma	11.8	17.4
	DeepSeek-Coder-v2	6.1	6.1
	Qwen2.5-Coder	22.9	27.0
	DeepSeek-R1	10.5	13.5
	Llama3.1	10.8	14.1
	Qwen2.5	19.2	24.1
Lenguaje Natural	CodeGemma	12.1	17.9
	DeepSeek-Coder-v2	12.0	14.6
	Qwen2.5-Coder	**24.4**	**28.3**
	DeepSeek-R1	18.1	23.7
	Llama3.1	14.8	19.0
	Qwen2.5	20.7	23.1

Tabla 5.3: Resultados de la evaluación para la tarea RE utilizando los dos tipos de instrucción en conjunto con los distintos modelos. Se muestra el promedio ponderado de la métrica F1 utilizando las dos estrategias de emparejamiento.

que no son necesariamente errores.

5.1.2. Comparación entre modelos

Se observa que para todas las tareas, el modelo Qwen2.5-Coder obtiene los mejores resultados. Sorprendentemente, para la tarea RE, incluso obtiene los mejores resultados utilizando la instrucción de lenguaje natural, lo cual contradice la intuición de que un Code-LLM obtendría mejores resultados cuando es utilizado con una instrucción de código. El hecho de que los mejores resultados hayan sido obtenidos por un Code-LLM ya había sido evidenciado por otros trabajos [4, 21, 36, 37, 44, 60, 76].

Este hecho puede deberse a que estos modelos están mejor capacitados para generar información estructurada debido a su vasto entrenamiento en corpus de código de programación. Es lógico pensar que la capacidad de generar salidas estructuradas no

Instrucción	Modelo	Exacta	Fuzzy
Código	CodeGemma	64.9	64.9
	DeepSeek-Coder-v2	62.2	62.2
	Qwen2.5-Coder	**69.5**	**69.5**
	DeepSeek-R1	30.0	30.0
	Llama3.1	51.3	51.3
	Qwen2.5	67.9	67.9
Lenguaje Natural	CodeGemma	36.4	36.4
	DeepSeek-Coder-v2	41.8	41.8
	Qwen2.5-Coder	54.5	54.5
	DeepSeek-R1	51.6	51.6
	Llama3.1	17.9	17.9
	Qwen2.5	51.4	51.4

Tabla 5.4: Resultados de la evaluación para la tarea ETD utilizando los dos tipos de instrucción en conjunto con los distintos modelos. Se muestra el promedio ponderado de la métrica F1 utilizando las dos estrategias de emparejado.

Instrucción	Modelo	Exacta	Fuzzy
Código	CodeGemma	35.2	36.7
	DeepSeek-Coder-v2	33.5	35.0
	Qwen2.5-Coder	**45.4**	**48.2**
	DeepSeek-R1	10.9	11.6
	Llama3.1	23.7	24.7
	Qwen2.5	42.7	45.7
Lenguaje Natural	CodeGemma	13.5	15.3
	DeepSeek-Coder-v2	16.2	18.2
	Qwen2.5-Coder	32.9	36.1
	DeepSeek-R1	27.8	32.6
	Llama3.1	12.0	12.5
	Qwen2.5	30.4	33.4

Tabla 5.5: Resultados de la evaluación para la tarea EAD utilizando los dos tipos de instrucción en conjunto con los distintos modelos. Se muestra el promedio ponderado de la métrica F1 utilizando las dos estrategias de emparejado.

sea la única habilidad requerida para completar las tareas de extracción de información; también se requiere de la capacidad de comprender el significado semántico del texto, resolver correferencias y realizar asociaciones entre entidades y valores. Es posible que los Code-LLM, al partir como NL-LLMs que luego fueron entrenados sobre grandes corpus de tareas de código, también sean capaces de resolver estas tareas del lenguaje natural, mientras que mantienen buenas propiedades de generación de información estructurada. Estas pueden ser las razones por las cuales el modelo Qwen2.5-Coder obtiene los mejores resultados.

El hecho de que Qwen2.5-Coder obtenga mejores resultados que los otros Code-LLM probados (CodeGemma y Deepseek-Coder-v2) no es sorprendente. CodeGemma es un modelo más pequeño y menos capaz, y Qwen2.5-Coder supera a Deepseek-Coder-v2 en la mayoría de las evaluaciones reportadas por los autores [86].

Es importante notar que comparando los NL-LLM, Qwen2.5 también es el que

Modelo	Inst.	NER	VE	RE	ETD	EAD	Prom.
CodeGemma	C	60.3	68.0	17.4	64.9	36.7	**49.5**
	LN	51.0	62.4	17.9*	36.3	15.3	36.6
Deepseek-Coder-V2	C	56.6	62.9	6.1	62.1	35.0	**44.5**
	LN	47.9	66.0*	14.6*	41.7	18.2	37.7
Qwen2.5-Coder	C	68.8	78.3	27.0	69.5	48.2	**58.4**
	LN	65.8	78.5*	28.3*	54.5	36.1	52.6
Llama3.1	C	46.6	60.5	14.1	51.2*	24.7*	**39.5**
	LN	63.5	72.1	19.0	17.9	12.5	37.0
Deepseek-R1	C	39.1	39.8	13.5	29.9	11.6	26.8
	LN	62.3	69.8	23.7	51.6	32.6	**48.0**
Qwen2.5	C	56.8	77.1*	24.1*	67.9*	45.7*	**54.3**
	LN	67.0	75.5	23.1	51.3	33.4	50.1
Promedio Modelos	C	54.7	64.4	17.0	57.6	33.6	**45.5**
	LN	59.6	70.7	21.1	42.2	24.7	43.7

Tabla 5.6: Comparación del uso de ambas instrucciones en el mismo modelo. La letra C representa la instrucción de código y LN la instrucción de lenguaje natural. Los valores mostrados son el promedio ponderado de la métrica F1 utilizando la estrategia Fuzzy. Los valores con marcados con * muestran los casos donde la instrucción utilizada no coincide con la naturaleza del modelo. La columna final muestra el promedio para todas las tareas. La última fila calcula el promedio de todos los modelos para cada instrucción.

obtiene los mejores resultados en todas las tareas, en contraposición a Llama3.1 y Deepseek-R1. Sobre Llama3.1, hay que destacar que la versión probada es más pequeña que las versiones de Qwen2.5 y Deepseek-R1 probadas. Sin embargo, el hecho de que Qwen2.5 obtenga mejores resultados puede deberse a que las capacidades del modelo de comprender el texto y resolver tareas de lenguaje natural básicas sean superiores a las de los demás modelos. A su vez, Qwen2.5 es el modelo de base utilizado para entrenar Qwen2.5-Coder en tareas de generación de código, lo cual explica por qué el segundo obtenga siempre resultados mejores que el primero. Es posible que Qwen2.5-Coder aproveche las habilidades de compresión del lenguaje natural de Qwen2.5, mientras que mejore las capacidades de generar información estructurada al entrenarse en tareas de generación de código. Esto último se observa claramente en la tabla 5.7, donde Qwen2.5-Coder obtiene porcentajes menores de salidas con errores, independiente de la instrucción utilizada, en comparación con Qwen2.5.

Por otro lado, los textos utilizados en este trabajo son relativamente cortos, la ontología de la información a extraer es relativamente acotada y el contenido de los textos suele ser bastante homogéneo. Esto puede significar que la dificultad de la tarea presente en este trabajo yace más bien en la generación de información estructurada y no tanto en las capacidades de comprender lenguaje natural. En los trabajos citados que evidencian este hallazgo, los textos utilizados para evaluar los sistemas suelen ser oraciones, lo cual también simplifica las habilidades de lenguaje natural necesarias para realizar la tarea. Sería interesante evaluar si el mismo resultado se mantiene si se trabaja con textos más extensos, que requieran de una comprensión más elevada, y donde la ontología de la información a extraer sea más compleja.

5.1.3. Comparación entre instrucciones

La tabla 5.6 permite contrastar los resultados obtenidos para ambas instrucciones.

Observando el promedio para todas las tareas (última columna), se concluye que en 5 de 6 casos la instrucción de código obtiene los mejores resultados, incluso cuando se utiliza en conjunto con NL-LLMs como Llama3.1 y Qwen2.5 que no están especializados en generar código. Esto evidencia la importancia de utilizar un lenguaje estructurado, como un lenguaje de programación, para instruir al LLM en la extracción de información estructurada.

El único modelo que obtiene mejores resultados en promedio con la instrucción de lenguaje natural es Deepseek-R1. Gracias a las capacidades de razonamiento y resolución de problemas de este modelo, es posible que esté mejor capacitado para comprender las reglas de extracción de la información presentes en la instrucción de lenguaje natural.

Mirando los resultados individuales para cada tarea y los promedios para todos los modelos (última fila), se observa que la instrucción de lenguaje natural es mejor para las tareas VE, NER y RE, mientras que la instrucción de código es mejor para las tareas ETD y EAD. En el caso de la tarea RE, es probable que instruir al modelo a generar sentencias que definen la relación (por ejemplo, "Juan Pérez es familiar de María Rodríguez") facilite la resolución de la misma en esta modalidad. Por otro lado, es posible que la descripción de la estructura de los eventos y sus argumentos sea demasiado compleja de realizar en lenguaje natural, por lo que resulte más fácil para los LLM entender los eventos y su estructura en formato código. Estas observaciones alientan a que se construya una estrategia mixta, que combine ejemplos de sentencias en lenguaje natural que se proveen dentro de la descripción en código de la información a extraer para la instrucción de código. Otra alternativa podría ser descomponer el sistema en componentes que resuelvan las tareas independientemente utilizando distintas instrucciones y modelos.

Los resultados marcados con asteriscos en la tabla señalan casos donde la instrucción que obtiene el mejor resultado no está alineada con el tipo de modelo utilizado. Si bien esto es cierto para la minoría de los casos, puede ser una señal de que la elección de tipo de instrucción y tipo de modelo son independientes, pudiéndose aprovechar un modelo potente de cualquier tipo y una instrucción de código para obtener buenos resultados. Sería interesante comprobar esta afirmación con NL-LLMs más grandes y potentes.

En esta línea, resulta interesante observar que para las tareas NER, RE y VE, la combinación de un Code-LLM con una instrucción de lenguaje natural obtiene iguales o incluso mejores resultados que utilizar una instrucción de código con estos modelos. Esto desafía la intuición planteada en etapas tempranas de la investigación de que un modelo de un tipo se comporta mejor con una instrucción del mismo tipo, y demuestra que pruebas cruzadas entre ambos tipos pueden ser beneficiosas para encontrar mejores combinaciones.

5.1.4. Análisis de errores en el Procesador de Salida

Como se describió en la sección 4.7, el módulo Procesador de Salida se encarga de verificar y procesar la salida del LLM con el objetivo de instanciar los modelos de datos definidos. Este procesamiento tiene dos objetivos: convertir la salida de texto del LLM a información estructurada que permita un procesamiento posterior eficiente y filtrar errores generados por el LLM. Debido a alucinaciones o falta de alineamiento con las instrucciones, el LLM puede generar salidas con errores dentro de alguna de estas categorías:

Instrucción	Modelo	% salidas con errores	Prom. de errores por salida
Código	CodeGemma	73.2	1.96
	DeepSeek-Coder-v2	53.4	**1.20**
	Qwen2.5-Coder	**45.5**	1.24
	DeepSeek-R1	**45.5**	1.56
	Llama3.1	64.3	2.10
	Qwen2.5	60.3	1.32
	Promedio	57.0	1.56
Lenguaje Natural	CodeGemma	82.1	1.46
	DeepSeek-Coder-v2	93.0	3.44
	Qwen2.5-Coder	67.3	1.71
	DeepSeek-R1	100.0	21.98
	Llama3.1	87.1	38.76
	Qwen2.5	71.2	1.58
	Promedio	83.45	11.70

Tabla 5.7: Comparación de los errores detectados por el Procesador de Salida para distintos modelos e instrucciones. Se visualiza el porcentaje de salidas (sobre el total de textos del conjunto de datos de evaluación) del LLM que tuvieron al menos un error, y la cantidad promedio de errores por salida del LLM. A su vez, se calcula el promedio para todos los modelos por instrucción.

- Categoría 1: Realizar modificaciones a la ontología de la información a extraer, agregando entidades, valores, relaciones, eventos o argumentos no definidos o renombrándolos.
- Categoría 2: No seguir la estructura definida para generar el texto con la información a extraer. En el caso de la instrucción de código, esto implicaría no seguir las reglas de instanciación de los modelos de datos o instanciarlos incorrectamente. En el caso de la instrucción de lenguaje natural, esto implicaría no seguir la estructura de los statements.
- Categoría 3: En el caso de las tareas RE y EAD, realizar referencias erróneas a entidades o valores no definidos o definidos erróneamente.
- Categoría 4: Agregar líneas innecesarias a la salida con comentarios u otras acotaciones a pesar de que se le haya especificado que no debe hacerlo en la instrucción.

Todos los errores generados por el LLM y detectados por el Procesador de Salida durante la evaluación de los modelos e instrucciones se almacenaron para realizar este análisis. La cantidad de errores se contabiliza siguiendo las reglas explicadas en la sección 4.7.3. En la tabla 5.7 se presenta una comparación del porcentaje de salidas del LLM que tienen al menos un error y la cantidad promedio de errores por salida del LLM para todas las combinaciones de modelo e instrucción.

Se observa que en promedio, la instrucción de código tiene un porcentaje de salidas con errores mucho menor que la instrucción de lenguaje natural. En el caso de los NL-LLMs como Deepseek-R1 y Llama3.1 esto puede deberse en gran parte a que estos LLMs agregan líneas innecesarias a la salida con explicaciones o comentarios (Categoría 4). En particular, Deepseek-R1 agrega una descripción de su razonamiento a cada salida con fragmentos como "*Looking at the example provided earlier, when a person is involved in an event like being transferred from one location to another, it's noted with specific relations and events. In this case, the text mentions people leaving Uruguay,*

so they are moving from there to another country (though not specified here)". Este tipo de fragmentos son descartados por el Procesador de Salida y contabilizados como errores, ya que se instruyó específicamente al LLM de no generar salidas adicionales a las requeridas para la tarea.

A modo de contabilizar y estudiar propiamente errores de las categorías 1, 2 y 3, se descartan todos los errores de la categoría 4. Esto nos deja con un total de 1189 errores en total para todos los modelos utilizando la instrucción de lenguaje natural y 951 errores en total para todos los modelos utilizando la instrucción de código. A su vez, se utiliza la herramienta Top2Vec [1] para analizar el texto de las excepciones para cada instrucción e intentar evaluar cuáles son las categorías más populares.

Para la instrucción de lenguaje natural, las 3 categorías más populares de errores son, en orden:

- Se generan statements para eventos que no siguen la estructura especificada (Categoría 2).
- Se agregan tipos de entidad o valor no existentes en la ontología (por ejemplo, "Reference", "Place" o "Thing") (Categoría 1).
- Se agregan argumentos de eventos que refieren a entidades que no fueron definidas antes (Categoría 3).

Para la instrucción de código, las 3 categorías más populares de errores son:

- Se agregan tipos de entidad o relación no existentes en la ontología (por ejemplo, "Abandon", "WorksFor" o "Contains") (Categoría 1).
- Se intentan crear relaciones o argumentos de eventos con los tipos equivocados (por ejemplo, intentar crear un argumento de tipo `origen` para un evento que apunte a una `Organización` y no un `Lugar`) (Categoría 2).
- Referir a un valor de nombre "Date" en vez de al definido en la ontología como `Time` (Categoría 1).

Un análisis más profundo de las categorías y subcategorías de los errores es útil para identificar los puntos de falla del sistema y encontrar puntos de mejora para las instrucciones o controles que se puedan agregar a la salida del LLM.

Es importante notar que, independientemente de la instrucción o modelo utilizado, las alucinaciones o desviaciones de las instrucciones aún ocurren. Esta deficiencia es principalmente responsabilidad del LLM, aunque ciertas estrategias se podrían implementar para intentar mitigar estos errores, como continuar refinando las instrucciones provistas o aplicar algún mecanismo de retroalimentación, donde los errores encontrados son retornados al LLM para que corrija su salida.

5.1.5. Observaciones sobre las tareas

Observando el promedio de los resultados para todos los modelos por tarea (última fila de la tabla 5.6), se destaca que la tarea que obtiene los valores más bajos es RE, seguida de EAD. Estas tareas presentan valores bajos para todos los modelos, independientemente de la instrucción utilizada. Ambas tareas requieren de las mismas habilidades subyacentes que implican asociar entidades o spans entre sí bajo una cierta premisa. Para las tareas NER, VE y ETD se observan mejores resultados en general. Estas tres tareas también comparten similitudes: todas requieren de la extracción y clasificación de spans bajo distintas premisas.

Es importante notar que las tareas RE y EAD sufren errores en cascada a partir de deficiencias en las tareas NER, VE y ETD. Ya que las tareas RE y EAD se encargan de emparejar entidades o valores extraídos por las demás tareas, cualquier error cometido

por estas puede provocar errores en las tareas RE y EAD. Por ejemplo, si la tarea de NER clasifica incorrectamente un span, cualquier relación que empareje ese span con otro estará también incorrecta. Esto hace que la evaluación para estas tareas sea injusta.

Una forma de hacer esta evaluación más justa podría ser utilizar las etiquetas correctas de las tareas NER, VE y ETD para las tareas RE y EAD en vez de dejar que el modelo las genere. Sin embargo, hoy en día las cinco tareas se resuelven en una única pasada por el LLM, y para lograr esta evaluación se requeriría descomponer el sistema en partes, lo cual implica un esfuerzo adicional que queda fuera del alcance de esta tesis. También, esta alternativa se aleja significativamente de la filosofía general del enfoque planteado en este trabajo, donde se intentan abordar todos los problemas en conjunto utilizando un único sistema.

A pesar de esta aclaración, estos resultados pueden aún estar evidenciando una deficiencia del sistema o los componentes utilizados en cuanto a las tareas RE y EAD. Dado que ambas tareas requieren de las mismas habilidades, se podrían explorar formas de instruir mejor al LLM en cuanto a como generar asociaciones entre spans.

5.2. Análisis del uso de ejemplos en la instrucción

En el diseño del sistema se incluyó un módulo opcional de Recuperación de Ejemplos, que se encarga de buscar ejemplos similares al texto de entrada en los datos de entrenamiento y formatearlos junto con la salida esperada para incluirlos en la instrucción. Esta técnica, que combina ICL y RAG, se utiliza para dar al modelo ejemplos útiles que le permitan entender la tarea y el formato de salida esperado.

En esta sección, se realiza un análisis del efecto de distintas cantidades de ejemplos en la instrucción sobre el conjunto de evaluación. Las figuras 5.1 y 5.2 muestran los resultados de la evaluación para todas las tareas, para distintos modelos e instrucciones, variando la cantidad de ejemplos dados en la instrucción. Debido a la extensa cantidad de tiempo requerida para ejecutar estas pruebas, se decidió acotar la cantidad de combinaciones de modelo e instrucción evaluadas. Se eligieron dos Code-LLM (CodeGemma y Qwen2.5-Coder) y dos NL-LLM (Llama3.1 y Deepseek-R1) representantes para estas pruebas, siendo dos de estos modelos de un tamaño de 5GB (CodeGemma y Llama3.1) y otros dos de un tamaño ligeramente mayor de 9GB (Qwen2.5-Coder y Deepseek-R1). También, solo se evaluaron los Code-LLM junto con la instrucción de código y los NL-LLM junto con la instrucción de lenguaje natural.

En primer lugar, se observa que para todas las configuraciones de modelo e instrucción evaluadas, el uso de algún número de ejemplos siempre es mejor que no utilizar ejemplos. Si se observa el promedio F1 para todas las tareas, se ve que el salto de no utilizar ejemplos a utilizar un solo ejemplo en la instrucción aumenta significativamente el promedio F1 para todas las tareas, para todos los modelos. Este incremento se ve acentuado para los modelos Code-LLM probados (Qwen2.5-Coder y CodeGemma), los cuales no son capaces de resolver ninguna tarea satisfactoriamente a menos que se les dé un ejemplo. Este hecho valida la utilidad del módulo de Recuperación de Ejemplos y sustenta las afirmaciones de trabajos previos que indican que el uso de ejemplos en la instrucción permite que los LLM puedan resolver con mayor desempeño tareas en modalidad few-shot.

En segundo lugar, se observa que para todos los modelos y tareas, el máximo valor de F1 se alcanza con 5 o menos ejemplos. En particular, para tres de las configuraciones modelo/instrucción probadas, el mejor promedio de F1 para todas las tareas se alcanza con $k = 3$. Esto puede deberse a que un mayor número de ejemplos incremente significativamente el largo de la instrucción, lo cual dificulte la capacidad del modelo

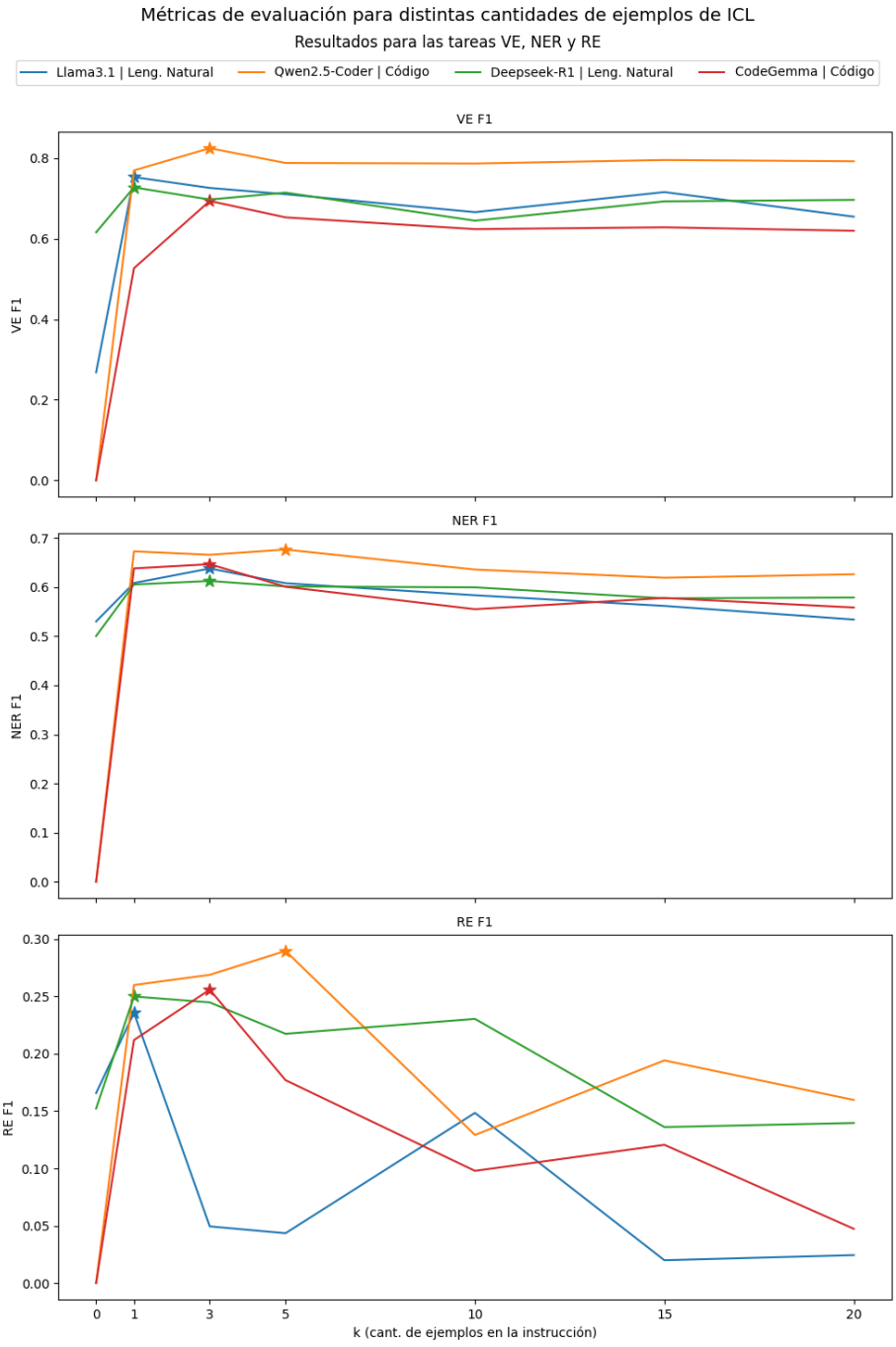


Figura 5.1: Comparación del efecto de utilizar distintas cantidades de ejemplos (k) en la instrucción para las tareas VE, NER y RE. Para cada tarea, se visualiza el promedio ponderado de la métrica F1 para distintos valores de k, utilizando diferentes modelos. Se marca con una estrella el valor de k en el que se obtiene el máximo valor de promedio ponderado de F1 para cada tarea y modelo.

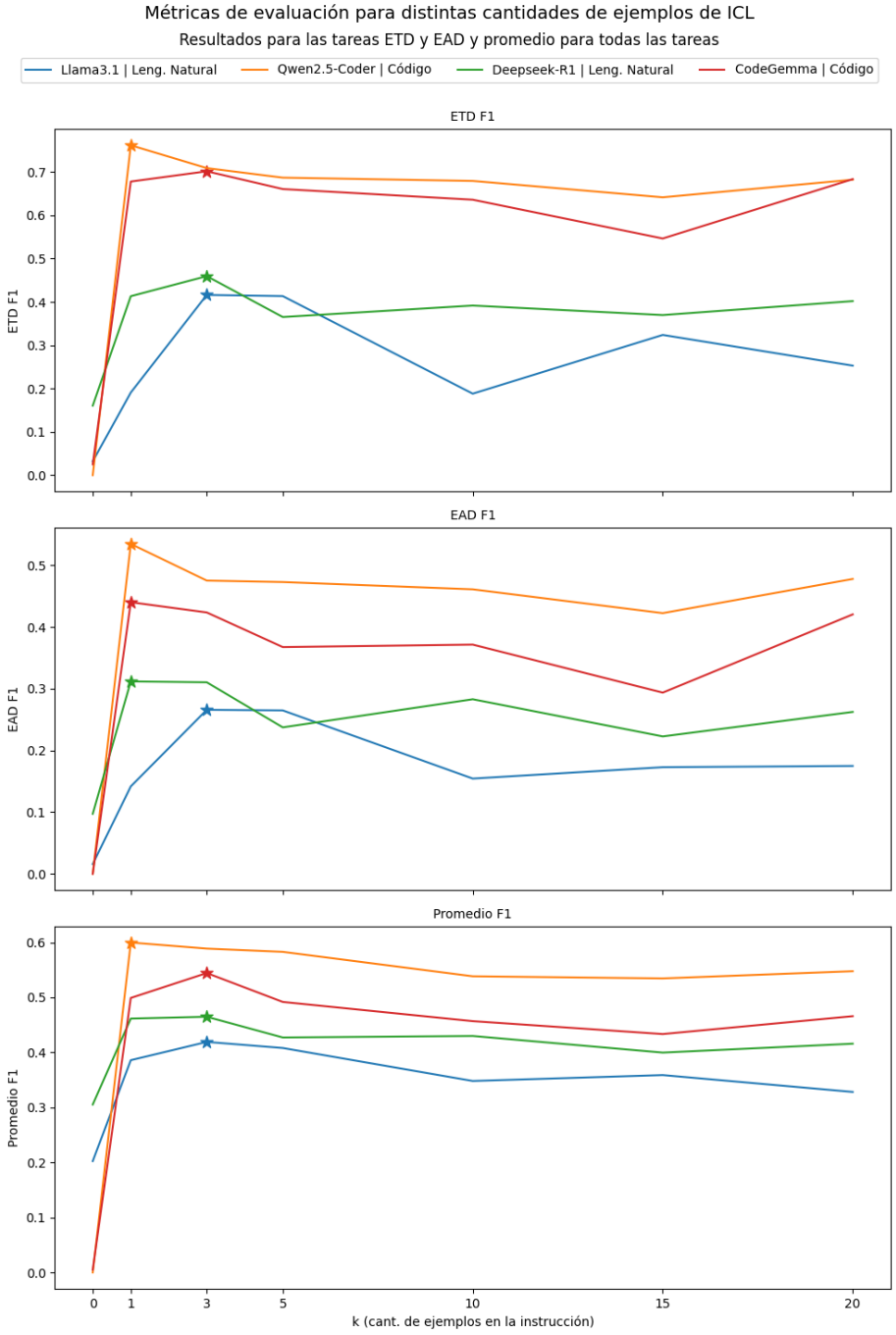


Figura 5.2: Comparación del efecto de utilizar distintas cantidades de ejemplos (k) en la instrucción para las tareas ETD y EAD. Para cada tarea, se visualiza el promedio ponderado de la métrica F1 para distintos valores de k, utilizando diferentes modelos. A su vez, se calcula el promedio de los valores F1 para todas las tareas. Se marca con una estrella el valor de k en el que se obtiene el máximo valor de promedio ponderado de F1 para cada tarea y modelo.

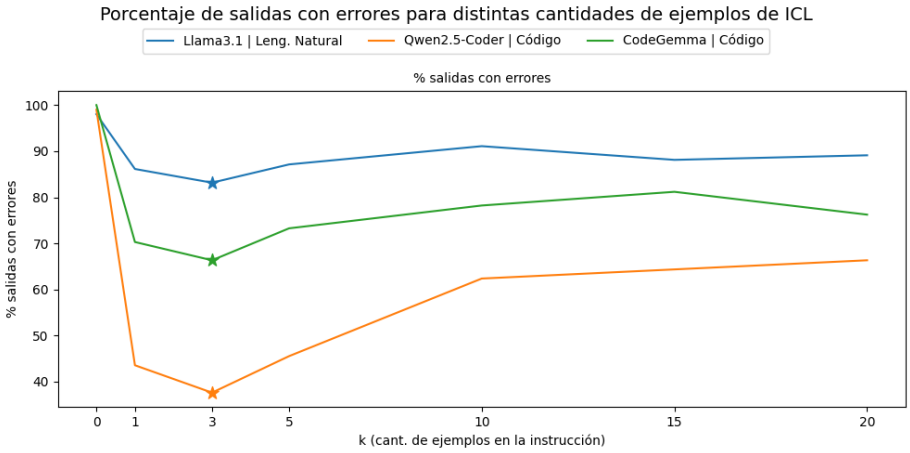


Figura 5.3: Comparación del efecto de utilizar distintas cantidades de ejemplos (k) en la instrucción sobre el porcentaje de salidas del LLM que tienen al menos un error. Se marca con una estrella el valor de k en el que se obtiene el menor porcentaje para cada modelo.

de procesarla completamente, teniendo resultados contraproducentes. Este hecho puede no ser cierto para modelos más grandes que los probados, que permitan procesar contextos de entrada más extensos. Para las configuraciones probadas, se concluye que no son necesarios más de 5 ejemplos para obtener buenos resultados.

A pesar de las variaciones en la calidad de los resultados dependiendo de k, es importante notar que las configuraciones que incluyen un Code-LLM junto con la instrucción de código suelen obtener mejores resultados con $k > 0$ que las configuraciones que incluyen un NL-LLM junto con la instrucción en lenguaje natural. Esto se observa con mayor claridad al ver el promedio de F1 para todas las tareas, donde los Code-LLM se encuentran por encima de los NL-LLM para todos los $k > 0$. Esto deja de ser cierto cuando $k = 0$, lo cual parece indicar que los NL-LLM junto con la instrucción en lenguaje natural son mejores para modalidad zero-shot, mientras que los Code-LLM junto con la instrucción de código son mejores para modalidad few-shot.

La figura 5.3 compara el porcentaje de salidas del LLM con errores para distintos valores de k. De este análisis se omite al modelo Deepseek-R1 con la instrucción de lenguaje natural, ya que produce errores de Categoría 4 (ver subsección 5.1.4) para todas las salidas al incluir su razonamiento.

Se observa que al igual que para los resultados de las métricas de evaluación, hay un salto importante en el porcentaje de salidas con errores entre no utilizar ejemplos ($k = 0$) y utilizar un solo ejemplo ($k = 1$). No utilizar ejemplos provoca que los tres modelos produzcan errores en todas las salidas, mientras que utilizar un solo ejemplo logra disminuir significativamente el porcentaje de salidas con errores. Este efecto es más notorio para el modelo Qwen-2.5-Coder con la instrucción de código, que con solo un ejemplo logra disminuir el porcentaje a menos del 50 %. La razón de por qué este modelo logra aprovechar mejor los ejemplos puede deberse a que este modelo es de un mayor tamaño que los otros dos evaluados (Llama3.1 y CodeGemma), lo cual permite que procese mejor contextos más largos y que pueda lograr prestar mayor atención al ejemplo dentro de la totalidad de la instrucción.

También se observa que el número óptimo de ejemplos para los tres modelos es $k = 3$, el mismo número óptimo de ejemplos para obtener los mejores resultados en promedio para todas las tareas. Al igual que antes, sería interesante evaluar si este k

óptimo crece si se utilizan modelos más grandes, con capacidad de procesar contextos más extensos. Por otro lado, se mantiene el hecho de que un $k > 3$ tiene efectos negativos y produce una mayor cantidad de salidas con errores.

5.3. Resolución implícita de correferencias

En la sección 2.4 se mencionó que, en etapas tempranas de este trabajo, se investigó el problema de resolución de correferencia y los algoritmos existentes para resolverlo, con el fin de proponer una solución que actuara en dos etapas: identificando menciones a entidades y las cadenas de correferencia que las unen, y luego utilizando esa información para resolver el problema de extracción de información. Durante el estudio del estado del arte se reveló que los LLM pueden atacar el problema de extracción de información directamente, y que son capaces de resolver correferencias de forma implícita. En esta sección, se describirán algunos ejemplos de fichas con correferencias y salidas del LLM donde estas correferencias se resuelven implícitamente. Para estos ejemplos, se configuró el sistema para que utilice Qwen2.5-Coder junto con la instrucción de código, $k = 3$ ejemplos de ICL y una temperatura de 0,1.

El ejemplo 5.1 muestra un texto de una ficha que incluye un caso de correferencia un tanto complejo. El nombre de la causante de la ficha, "Julia SUAREZ", se menciona solo al comienzo de la ficha. Luego, en el texto, se hace uso del sujeto omitido para aportar más información sobre la causante. La oración se podría reescribir incluyendo el sujeto de la siguiente forma: "El 15.1.74 se establece que [Julia SUAREZ] es socia del Mov. Coord. de Magisterio.-". En estos casos, se dice que se tiene una relación de correferencia entre la mención "Julia SUAREZ" y el sujeto omitido (representado por un span vacío).

Ejemplo 5.1: Ejemplo de ficha que contiene correferencias. En este caso, la correferencia se da entre la mención "Julia SUAREZ" y un sujeto omitido en el texto, ambos marcados con corchetes rectos.

```
Causante: [Julia SUAREZ]

Nro. Ficha: 21996

Fecha: None

Texto: ANTECEDENTES: El 15.1.74 se establece que [] es socia
    ↪  del Mov. Coord. de Magisterio.-
```

El ejemplo 5.2 muestra la salida cruda del LLM (sin pasar por el Procesador de Salida) para el texto de entrada del ejemplo 5.1. Podemos observar que el sistema correctamente identifica dos entidades (la `Persona` "Julia SUAREZ" y la `Organización` "Mov. Coord. de Magisterio") y una relación `Integra` entre ambas entidades. Esta relación sería imposible de identificar si no se resolviera implícitamente el sujeto omitido y la mención correferente en la oración, ya que la oración no menciona directamente a "Julia SUAREZ".

Ejemplo 5.2: Salida cruda del LLM (sin pasar por el Procesador de Salida) para la ficha de entrada del ejemplo 5.1.

```
# Entities
entity1 = Person(span='Julia SUAREZ', other_spans=[])
entity2 = Organization(span='Mov. Coord. de Magisterio', other_spans=[])
```

```
# Values
value1 = Time(span='15.1.74')

# Relations
relation1 = Integrates(gov=entity1, dep=entity2)
```

El ejemplo 5.3 muestra un caso un poco más complejo e interesante. Este texto contiene varias correferencias, siendo una de estas la correferencia entre la mención "del requerido DANIEL PEDRO ALFARO VAZQUEZ (a) Alejandro" y la mención "requerido", ambas marcadas con corchetes rectos en el texto. Es importante notar que resolver esta correferencia en el texto es clave para identificar que la mención "requerido" refiere a la entidad Alejandro Daniel Pedro Alfaro Vaquez, y no a otra de las entidades mencionadas en el texto, como Fernando Guzman Alfaro Vazquez o Luis Enrique.

Ejemplo 5.3: Ejemplo de ficha que contiene correferencias. En este caso, una de las correferencias se da entre la mención "del requerido DANIEL PEDRO ALFARO VAZQUEZ (a) Alejandro" y la mención "requerido". Ambas se encuentran señaladas con corchetes rectos en el texto.

```
Causante: FERNANDO GUEZMAN ALFARO VAZQUEZ

Nro. Ficha: 0776

Fecha: 19-11-73

Texto: ANTECEDENTES: Era hermano [del requerido DANIEL PEDRO
    ↪  ALFARO VAZQUEZ (a) Alejandro], el cual se intentaba
    ↪ detener en el procedimiento en la chacra finca, en
    ↪ cual se encontraba vinculado al caso del sedicioso ya
    ↪ capturado LUIS ENRIQUE.- Se puso comprobar que dicho [
    ↪ requerido] se encontraba en Buenos Aires, viviendo en
    ↪ el HOTEL CENTRAL, calle CANGALLA 1157.- Sera requerido
    ↪  por esta Unidad. Seg n ficha de PROCEDIMIENTO DE BN
    ↪ .1 No. 6885 del d a 7-11-73.-
```

El ejemplo 5.4 muestra la salida cruda del LLM para el texto de entrada del ejemplo 5.3. Se puede observar que el sistema identifica correctamente las entidades `Persona` "DANIEL PEDRO ALFARO VAZQUEZ (a) Alejandro" y `Lugar` "HOTEL CENTRAL", y la relación `domicilio` entre ambas entidades. Esta relación sería imposible de identificar correctamente sin resolver la correferencia entre la mención "requerido" y la mención "del requerido DANIEL PEDRO ALFARO VAZQUEZ (a) Alejandro", ya que permite desambiguar a quién refiere dicha mención de entre los distintos nombres mencionados en el texto.

Ejemplo 5.4: Salida del LLM (sin pasar por el Procesador de Salida) para la ficha de entrada del ejemplo 5.3.

```
# Entities
entity1 = Person(span='FERNANDO GUEZMAN ALFARO VAZQUEZ', other_spans
    ↪ =[])
entity2 = Person(span='DANIEL PEDRO ALFARO VAZQUEZ (a) Alejandro',
    ↪ other_spans=[])
entity3 = Location(span='Buenos Aires', other_spans=[])
entity4 = Location(span='HOTEL CENTRAL', other_spans=[])
entity5 = Location(span='calle CANGALLA 1157', other_spans=[])
```

```
entity6 = Person(span='LUIS ENRIQUE', other_spans=[])

# Values
value1 = Time(span='19-11-73')
value2 = Time(span='7-11-73')

# Relations
relation1 = IsFamily(gov=entity1, dep=entity2)
relation2 = LivesIn(gov=entity2, loc=entity4)

# Events
event1 = Detention(trigger='intentaba detener', subject=[entity2], dst=
    ↪ None, time=value1, cause=[None])
event2 = Detention(trigger='capturado', subject=[entity6], dst=None,
    ↪ time=None, cause=[None])
```

Estos ejemplos permiten visualizar la capacidad de los LLM de resolver las correferencias de forma implícita al trabajar en tareas aguas abajo que requieran la desambiguación de menciones a entidades. Esta capacidad permite utilizar LLMs para atacar el problema de extracción de forma directa, sin la necesidad de pasos previos que resuelvan las correferencias en el texto.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Conclusiones y Trabajo a Futuro

En este capítulo se presentan las conclusiones obtenidas a lo largo del trabajo y se proponen posibles líneas de trabajo a futuro.

6.1. Transcripción de las fichas

Si bien la transcripción de las fichas no era el objetivo principal de este trabajo, este es un paso vital para extraer el texto de las imágenes previo a procesarlo utilizando algoritmos de PLN para diferentes tareas. La calidad de la transcripción también es importante para que tareas aguas abajo puedan cumplirse de forma adecuada.

En este trabajo se realizó fine-tuning del modelo multi-modal MiniCPM-Llama3-V2.5 con un conjunto de 174 fichas transcriptas manualmente para lograr transcribir todo el conjunto de partes traseras de las fichas de la OCOA. Las salidas del modelo luego fueron curadas a través de un proceso de selección y corrección manual de las transcripciones de las fichas, para obtener un subconjunto de fichas y transcripciones de alta calidad. En esta labor, el modelo entrenado fue de gran ayuda para acelerar el proceso de selección y corrección, y se observó que la transcripción cruda del modelo era lo suficientemente buena en la mayoría de los casos.

La transcripción del resto de los documentos del archivo Berrutti sigue siendo un problema de alta dificultad debido a la heterogeneidad de los documentos y la falta de calidad en los escaneos. La observación más importante a realizar de la etapa de transcripción de este trabajo es la factibilidad de realizar fine-tuning de un LLM multi-modal para obtener transcripciones de calidad de las imágenes. Una línea interesante a explorar es expandir el fine-tuning de este modelo hacia otros tipos de documentos.

Como salida de la etapa de transcripción de las fichas, y gracias a la asistencia del modelo generado, se obtuvo un conjunto de datos de 1000 transcripciones de fichas, seleccionadas y corregidas manualmente. Este conjunto es de utilidad para futuros trabajos que intenten mejorar la transcripción de estas fichas u otros documentos del archivo Berrutti, y se considera un aporte adicional de este trabajo.

En otra línea, los avances recientes en modelos de lenguaje multi-modales, como el modelo MiniCPM-Llama3-V2.5, sugieren la posibilidad de saltar el paso de transcripción de los documentos enteramente. Como se observó en los experimentos realizados, el modelo es capaz de estructurar la transcripción de la ficha según secciones presentes en la imagen de esta, extrayendo información como el número de ficha o el nombre y apellido del causante. Es interesante explorar si la tarea de extracción de información se puede resolver de forma completa al intentar aplicarla directamente sobre la imagen de la ficha y no sobre el texto transcripto. Si este enfoque lograra buenos resultados,

se podría acelerar el procesamiento de los documentos al evitar la transcripción de los textos e incluso beneficiar los resultados obtenidos en la tarea IE con información que se encuentra presente en la imagen pero no en el texto transcripto (por ejemplo, la estructura de la ficha o la posición de ciertas secciones de texto).

6.2. Creación del conjunto de datos

Como parte de este trabajo se llevó a cabo una tarea de etiquetado que resultó en un conjunto de datos anotado sobre las fichas de la OCOA para las tareas de extracción de entidades, relaciones y eventos. El conjunto de datos final cuenta con 403 fichas anotadas. Se espera que este conjunto de datos sea de utilidad para trabajos futuros que intenten profundizar en la extracción de información sobre los documentos del pasado reciente.

En la tarea de etiquetado participó un grupo interdisciplinario de anotadores pertenecientes a las carreras de Ciencias Sociales e Ingeniería en Computación. Contar con un grupo heterogéneo de anotadores con distintos conocimientos previos resultó enriquecedor para el proceso; algunos estudiantes aportaron mayor conocimiento del dominio a la tarea, el cual resultó útil para desambiguar menciones a ciertas entidades; otros estudiantes aportaron críticas importantes al proceso de etiquetado, ayudando a refinarlo iterativamente.

En la evaluación de la calidad de las anotaciones se pudo observar que las tareas con peores valores para las métricas de acuerdo entre anotadores fueron las tareas Relation Extraction y Event Argument Detection, que comparten características similares: ambas consisten en enlazar pares de spans extraídos y en clasificar este enlace. Si bien estas tareas se ven afectadas por errores en cascada de las tareas Named-Entity Recognition, Value Extraction y Event Trigger Detection, los bajos valores de acuerdo pueden indicar que estas tareas no fueron correctamente explicadas en la guía, o que se requiere de mayor asistencia para los anotadores en cuanto a estas tareas.

Las críticas y aprendizajes tomados de la tarea de etiquetado son vitales para mejorar el proceso en nuevas tareas de etiquetado que puedan llevarse a cabo dentro del proyecto CRUZAR.

6.3. Extracción de información

En este trabajo se construyó un sistema de extracción de información para las fichas de la OCOA. En la construcción del sistema se experimentó con distintas configuraciones de los componentes del sistema, con el objetivo de obtener una configuración que obtuviera buenos resultados y validara observaciones realizadas por trabajos previos.

Reflexiones sobre el tipo de LLM

En los experimentos realizados, el Code-LLM Qwen2.5-Coder fue el que obtuvo los mejores resultados. El hecho de que un LLM especializado en comprender y generar código obtenga los mejores resultados en una tarea que requiere fuertes habilidades de procesamiento del lenguaje natural ya había sido evidenciado por otros trabajos. Este modelo utiliza como base el NL-LLM Qwen2.5, que también es el mejor modelo de los NL-LLM probados. Es posible que Qwen2.5-Coder aproveche las habilidades de procesamiento de lenguaje natural de su modelo base, Qwen2.5, y mejore su capacidad de producir salidas estructuradas, la cual es una habilidad necesaria e importante para la tarea de extracción de información. Como se ha mencionado en otros trabajos, es posible que los Code-LLM estén mejor equipados para producir salidas estructuradas

debido a su extenso entrenamiento sobre corpus de código escritos en lenguajes de programación, que son altamente estructurados. Esto también se evidencia en los resultados del análisis de errores producidos por el LLM, donde Qwen2.5-Coder produce siempre un porcentaje de salidas con errores menor a Qwen2.5, señalando que el primero está mejor equipado para cumplir con las reglas de formato y salida estructurada de la tarea en cuestión.

Reflexiones sobre el tipo de instrucción

En la construcción del sistema se diseñaron dos instrucciones, una de código y otra de lenguaje natural, que describen las tareas de extracción de información y la información que se debe extraer con el fin de instruir al LLM. En promedio, la instrucción de código fue la que obtuvo mejores resultados. Esta instrucción fue diseñada tomando inspiración de trabajos previos, y su objetivo es modelar el problema de extracción de información como uno de instanciación de clases de Python. Al modelar el problema de esta forma, se aprovecha la estructura inherente del lenguaje de programación para representar la información que se desea extraer y la capacidad del LLM de comprender y operar sobre dichas estructuras para generar la salida.

El hecho de que, en promedio, los mejores resultados sean obtenidos por un Code-LLM en conjunto con una instrucción de código resulta interesante y podría evaluarse en otros problemas de PLN o aplicaciones que impliquen la generación de salidas estructuradas. Sin embargo, el conjunto de datos elegido para este trabajo es relativamente simple debido a su acotado dominio y la extensión de los documentos. Sería interesante evaluar si estas conclusiones se mantienen sobre documentos más extensos, o cuyo dominio de información a extraer sea más amplio o complejo.

Si bien la instrucción de código obtiene los mejores resultados en promedio, independiente de si se utiliza un Code-LLM o un NL-LLM, la instrucción de lenguaje natural obtiene mejores resultados para las tareas NER, VE y RE promediando para todos los modelos. El hecho de no tener un consenso absoluto sobre cuál es la mejor instrucción para todas las tareas sugiere que una estrategia mixta podría ser interesante de explorar en el futuro. Una forma de realizar esto podría ser mantener la descripción de la estructura de la información a extraer como clases de Python, pero agregando descripciones de las clases y sus atributos más ricas en lenguaje natural como comentarios en el código entregado al LLM. Otra forma, simulando la estrategia de Chain-of-Thought prompting [79], sería solicitar al modelo que acompañe la instanciación de los objetos de información a extraer con comentarios en lenguaje natural que expliquen la acción que se está tomando mediante código.

Reflexiones sobre los errores del LLM

Del análisis de errores en el Procesador de Salida se destaca la importancia de validar las salidas producidas por el LLM. Debido a alucinaciones o desalineamiento con las instrucciones dadas, el LLM puede generar salidas que no cumplen las reglas establecidas en la instrucción o que no se ajustan a la estructura de la información a extraer. La instrucción de código es la que obtiene la menor cantidad de errores en promedio, posiblemente debido a que las estructuras de la información a extraer son más fáciles de comprender al estar escritas en lenguaje de programación y son más fáciles de generar por el modelo mediante instancias de clases. Sin embargo, esta instrucción también produce salidas con errores, independiente de con qué modelo se utilice. El porcentaje más bajo de salidas con errores es 45.5 % y se obtiene con la instrucción de código y los modelos DeepSeek-R1 y Qwen2.5-Coder. Esto quiere decir que, incluso en el mejor caso, en aproximadamente la mitad de los ejemplos de evaluación, el LLM produjo al menos un error en la salida. Esto nos permite concluir

que aún hay espacio para la mejora en cuanto a asegurar que el LLM no produzca salidas indeseadas.

En este análisis de errores también se pudo observar que algunos de los errores más frecuentes implican que el LLM inventa entidades, relaciones o eventos que no fueron explicitados dentro de la información a extraer. Sin embargo, en algunos casos, los tipos inventados por el LLM refieren a información que podría ser de interés para investigadores pero que no fue explícitamente definida en el dominio de la información a extraer. Por ejemplo, el valor "Reference", que apareció en múltiples de las salidas obtenidas con la instrucción de lenguaje natural, y el cual aparentemente alude a documentos de referencia que aparecen mencionados dentro de la ficha (por ejemplo, "... Ver documentos (No. 360) del G.A.1..."), es un valor que podría ser útil para identificar otras fuentes de información que permitan validar eventos o relaciones. Lo que es un error en la presente tarea, podría ser una funcionalidad interesante de los LLM para atacar otras tareas como la extracción de información de dominio abierto (OIE, por su nombre en inglés, *Open Information Extraction*), donde se pretende que el sistema extraiga información estructurada a partir del texto sin indicaciones explícitas de qué información debe extraer.

Reflexiones sobre las tareas

Tanto en el acuerdo entre anotadores como en los resultados de la experimentación, los peores resultados fueron obtenidos para las tareas RE y EAD. Previamente, se explicó que los valores de las métricas para estas tareas se ven afectados por errores en cascada de las demás tareas (NER, VE y ETD). Para un análisis más justo de los resultados por tarea, sería útil forzar a que el sistema utilice las etiquetas del gold standard para las tareas NER, VE y ETD e intente resolver las tareas RE y EAD con las entidades, valores y disparadores de eventos correctos. Debido al esfuerzo que requeriría dicha prueba, se decidió dejarla fuera del alcance de este trabajo, aunque sería interesante incluir ese tipo de estudio en futuras iteraciones.

A pesar del posible efecto de los errores en cascada sobre los valores de las métricas para RE y EAD, las apreciaciones personales de los anotadores y del autor identifican ambas tareas como las más difíciles de anotar durante la tarea de etiquetado. Esto también puede indicar que estas tareas sean las más complejas de resolver por el sistema de extracción de información. Ambas tareas implican el enlazado de entidades o valores y la clasificación de dicho enlace.

En cuanto a la tarea de etiquetado, para mejorar el acuerdo entre anotadores se debe reforzar la guía de etiquetado en las secciones referentes a las tareas RE y EAD, así como también mejorar el entrenamiento de los anotadores para estas tareas. Por otro lado, podría ser interesante plantear un sistema de sugerencia de anotaciones con el fin de asistir a los anotadores en estas tareas, dada la efectividad del sistema de sugerencias provisto por INCEpTION para las tareas NER, VE y ETD.

En cuanto al sistema de extracción de información, es necesario explorar formas de mejorar los resultados para las tareas RE y EAD. Del análisis de distribución de las etiquetas del conjunto final realizado en la sección 3.4.3, se observa que la cantidad de etiquetas por clase para las tareas RE y EAD es relativamente menor que para las tareas NER y VE. Contar con un mayor número de etiquetas por clase podría permitir realizar fine-tuning de los LLM con el objetivo de mejorar los resultados en general.

Por otro lado, podría ser útil implementar un mecanismo de reintento para los casos en los cuales el LLM produce relaciones o eventos que no siguen la estructura planteada. Utilizando la lista de errores obtenida por el módulo de Procesador de Salida, se podría proveer dicha lista al LLM en conjunto con la instrucción original, indicándole que debe corregir su salida en base a la retroalimentación obtenida por el

Procesador de Salida. De esta forma, se le da al LLM una directiva implícita de qué acciones son incorrectas, lo cual podría ayudar a reducir el porcentaje de salidas con error y mejorar los resultados para las tareas RE y EAD, que requieren de estructuras más complejas.

En el futuro también resultaría útil realizar un análisis más detallado de los resultados por cada clase de cada una de las tareas. Con los valores de las métricas discriminados por clase, se podrían hacer ajustes más específicos de las instrucciones para cada clase, e identificar los tipos de entidad, valor, relación o evento más problemáticos.

Reflexiones sobre el uso de ejemplos

El uso de ejemplos demostró ser fundamental, no solo para obtener mejores resultados en las tareas sino también para disminuir el porcentaje de salidas con errores. Para el modelo Qwen2.5-Coder, pasar de no utilizar ejemplos a utilizar un solo ejemplo en la instrucción logró disminuir el porcentaje de salidas con errores en más de un 50 %. El uso de ejemplos es clave para demostrar al LLM cómo debe generar la salida estructurada y cómo resolver la tarea solicitada.

Para los LLM utilizados, los mejores resultados en promedio fueron obtenidos con menos de 5 ejemplos, lo cual implica que no se requiere de un extenso conjunto de datos etiquetados para poder aplicar la técnica. Esto refuerza el hecho de las capacidades de utilización de los LLM en modalidad few-shot: con un volumen pequeño de datos etiquetados y sin realizar entrenamiento, se pueden utilizar los ejemplos para lograr mejoras significativas en los resultados.

Si se utilizan más de 5 ejemplos, los resultados suelen ser contraproducentes. Esto puede deberse a que la instrucción se extiende demasiado en tamaño, lo cual dificulta su procesamiento adecuado por los LLM utilizados en estas pruebas. Sería interesante evaluar si esto se mantiene al utilizar LLMs de mayor tamaño y con la capacidad de procesar instrucciones y contextos de mayor longitud, donde puedan aprovechar un mayor número de ejemplos diferentes para cumplir la tarea.

Reflexiones sobre la resolución implícita de correferencias

Durante la experimentación se logró validar que los LLM cuentan con la capacidad de resolver las correferencias de forma implícita, lo cual permite la construcción de un sistema más sencillo que no utilice un paso explícito de resolución de correferencias.

Sin embargo, trabajos previos incorporan la información de los enlaces de correferencia durante el entrenamiento de distintos modelos, con el objetivo de mejorar los resultados para las tareas de extracción de información. Dado que el conjunto de datos etiquetado cuenta con información de correferencias entre menciones, podría ser interesante en el futuro incorporar dicha información al modelo de forma explícita y analizar su efecto sobre los resultados finales.

Reflexiones sobre los resultados en general

En todos los experimentos realizados, la mejor configuración obtuvo un promedio ponderado de la métrica F1 de 58.4 (utilizando la estrategia de emparejamiento Fuzzy), promediado sobre todas las tareas. Esto implica que aún hay espacio para la mejora en los resultados generales del sistema construido.

Todos los LLM utilizados son de pequeño tamaño (menos de 10GB) y podrían correr en GPUs de uso general. Si bien estos experimentos permitieron validar que es viable obtener una solución funcional con modelos de un tamaño menor a 10GB y

escasos recursos de hardware, podría ser interesante evaluar el comportamiento del sistema con modelos de mayor tamaño y con mayores capacidades. Para esto, es necesario disponer de mayores recursos de hardware.

En trabajos previos, se demostró que el fine-tuning de los modelos es efectivo para mejorar los resultados. En el futuro, extender el conjunto de datos con una mayor cantidad de documentos etiquetados y un mayor dominio de información a extraer podría propiciar el uso de estrategias de fine-tuning, en conjunto con las ya exploradas en este trabajo, para obtener un sistema más robusto y de alto rendimiento.

Apéndice A

Transcripción de las fichas

Previo a la extracción de información de los textos de las fichas, se debe aplicar una solución de transcripción automática (OCR, por su nombre en inglés, *Optical Character Resolution*) para extraer el texto de la imagen de la ficha. Estas imágenes cuentan con texto escrito mediante máquina de escribir o escrito a mano, en cursiva o imprenta.

En etapas tempranas de la investigación se probaron modelos como Tesseract [62], EasyOCR [1] y MiniCPM [87]. Tesseract e EasyOCR poseen ambos una arquitectura basada en redes neuronales profundas entrenadas para la tarea de OCR. MiniCPM es un LLM multi-modal, capaz de trabajar con texto e imágenes, que cuenta con 8.54 billones de parámetros, y que fue especializado en la tarea de OCR. Los tres modelos fueron evaluados en modalidad zero-shot para extraer el texto de la imagen. De estos, solo MiniCPM obtuvo resultados alentadores, pero aún así la calidad de la transcripción era insuficiente para los requerimientos del trabajo de extracción de información.

Debido a que el lenguaje y vocabulario utilizado en las fichas es bastante homogéneo, se realizó un experimento de fine-tuning del modelo MiniCPM-Llama3-V2.5 [87] con 50 fichas transcriptas a mano y evaluando sobre 30 fichas, comparando contra el modelo sin fine-tuning. Al modelo obtenido de esta etapa preliminar de fine-tuning se lo llama MiniCPM-FT y al modelo previo al fine-tuning simplemente MiniCPM. Resultados prometedores del modelo MiniCPM-FT propiciaron la realización de una etapa de fine-tuning más extensa.

A.1. Fine-tuning de MiniCPM

Para esta etapa, se transcribieron manualmente 214 fichas: 174 para entrenamiento y 40 para evaluación del modelo entrenado. Este conjunto de datos de evaluación no se intersecta con el conjunto de entrenamiento de 50 fichas de la etapa previa, y se utiliza luego para comparar todas las variantes de modelos probadas.

El objetivo principal de la tarea es la transcripción de las anotaciones que contienen información sobre antecedentes o datos sobre el sujeto/a de la ficha. Sin embargo, la ficha también contiene un número de ficha que ayuda a indexarla en el archivo de la OCOA, y a su vez puede contener los nombres y apellidos de la persona, la fecha en la que fue rellenada la ficha y referencias útiles. Aprovechando la capacidad del modelo MiniCPM-Llama3-V2.5 no solo de transcribir el texto sino que también de hacerlo

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

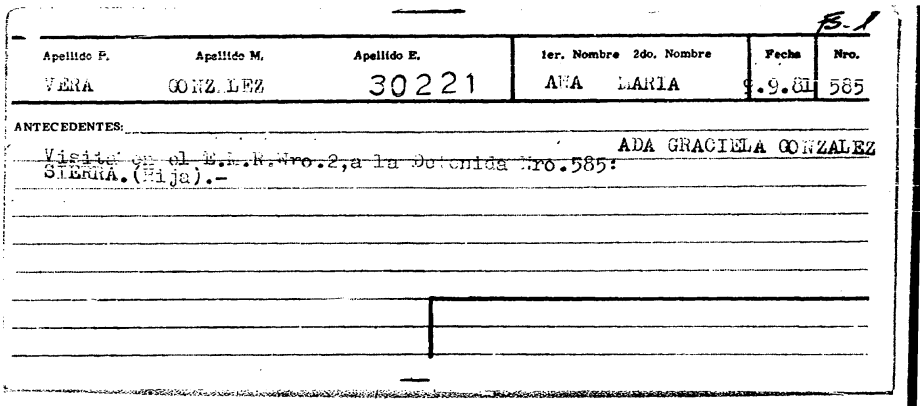


Figura A.1: Ficha de ejemplo.

de forma estructurada, las transcripciones utilizadas para entrenar el modelo extraen toda esta información de ser posible. El ejemplo A.1 muestra la transcripción y su formato para la ficha de ejemplo de la figura A.1.

Ejemplo A.1: Ejemplo de transcripción para la ficha de la figura A.1.

```
Nombres: ANA MARIA

Apellidos: VERA GONZALEZ

Nro. Ficha: 30221

Fecha: 9.9.81

Texto: ANTECEDENTES: Visita en el E.M.R.Nro.2, a la
Detenida Nro. 585: ADA GRACIELA GONZALEZ SIERRA. (Hija).

Referencias: None
```

Si uno de los campos `Nombres`, `Apellidos`, `Nro. Ficha`, `Fecha` o `Referencias` no se encuentra en la ficha, se rellena con el valor `None`.

El modelo resultante de esta etapa de fine-tuning con un mayor volúmen de datos se llama MiniCPM-FT-Ext por fine-tuning extendido.

A.1.1. Resultados obtenidos

Para analizar la efectividad de las etapas de fine-tuning, se comparan los modelos MiniCPM, MiniCPM-FT y MiniCPM-FT-Ext sobre el conjunto de datos de evaluación de 40 fichas. Se utilizan las siguientes métricas para evaluación:

- Puntaje diccionario: porcentaje de las palabras de la transcripción que se encuentran dentro de un diccionario común de palabras en español. Esta métrica es utilizada en otros proyectos de CRUZAR para evaluar la calidad de las transcripciones.
- Character Error Rate (CER): cantidad de errores en la transcripción a nivel de caracteres, comparado contra una transcripción deseada. Se cuentan la cantidad

	Puntaje Diccionario ↑	CER ↓	WER ↓
MiniCPM	73.19	4.31	5.0
MiniCPM-FT	**89.7**	3.23	4.46
MiniCPM-FT-Ext	85.92	**1.62**	**2.55**

Tabla A.1: Comparación de los resultados obtenidos por el modelo de transcripción en las tres etapas de fine-tuning. Se muestra el promedio de las métricas sobre todos los ejemplos del conjunto de datos de evaluación.

de inserciones, eliminaciones y substituciones de caracteres que se deben realizar en la transcripción obtenida para convertirla a la deseada y se divide sobre el total de caracteres. Un valor menor es mejor.

- Word Error Rate (WER): cantidad de errores en la transcripción a nivel de palabras, comparado contra una transcripción deseada. Se cuentan la cantidad de inserciones, eliminaciones y substituciones de palabras que se deben realizar en la transcripción obtenidas para convertirla a la deseada y se divide sobre el total de palabras. Un valor menor es mejor.

La tabla A.1 detalla los resultados de la evaluación para los tres modelos.

En primer lugar, se observa que el puntaje diccionario es mejor para la versión MiniCPM-FT que la versión MiniCPM-FT-Ext. Esto no es necesariamente malo. Muchas de las palabras presentes en las fichas transcriptas son siglas o abreviaciones que pueden no encontrarse directamente en el diccionario de palabras en español utilizado. Durante la construcción del sistema, se notó que el modelo tiende a convertir abreviaciones o siglas a palabras comunes del español, lo cual aumenta el puntaje diccionario pero es inherentemente malo para la transcripción. Lo importante es notar que el fine-tuning logra subir más de 10 puntos porcentuales el puntaje diccionario en comparación con la versión sin fine-tuning, lo cual es un indicio de que muchas palabras se están transcribiendo de forma distinta.

Para evaluar mejor la calidad de los modelos, es importante ver los valores de las métricas CER y WER. Ambas etapas de fine-tuning logran mejorar los valores de CER y WER, obteniendo MiniCPM-FT-Ext los mejores resultados. Un valor de 2.55 de WER indica que, del total de palabras de cada ficha, en promedio suelen haber entre 2 y 3 palabras con error de transcripción, lo cual es una gran mejoría en comparación con el WER de 5.0 obtenido por el modelo MiniCPM. A su vez, el valor de CER de 1.62 indica que del total de caracteres, en promedio suelen haber entre 1 y 2 caracteres con error versus entre 4 y 5 para el modelo MiniCPM.

En base a estos resultados, se elige el modelo MiniCPM-FT-Ext para realizar la transcripción completa del conjunto total de fichas disponibles, el cual cuenta con aproximadamente 22 mil fichas. De estas, se seleccionaron manualmente 1000 fichas para etiquetar como parte de la tarea de extracción de información.

Es importante destacar que estos resultados también indican que aumentar el volumen de datos y el tiempo de entrenamiento en la etapa de fine-tuning logra mejores resultados. En el futuro, se podría explorar aumentar el conjunto de datos etiquetados o aplicar esta misma estrategia de fine-tuning del modelo MiniCPM para otros tipos de documentos del archivo CRUZAR.

A.1.2. Comparación con métodos previos

Para visualizar la mejora de calidad obtenida por el preprocesamiento realizado en [49] y el modelo conseguido en esta etapa, se compara la salida de la transcripción

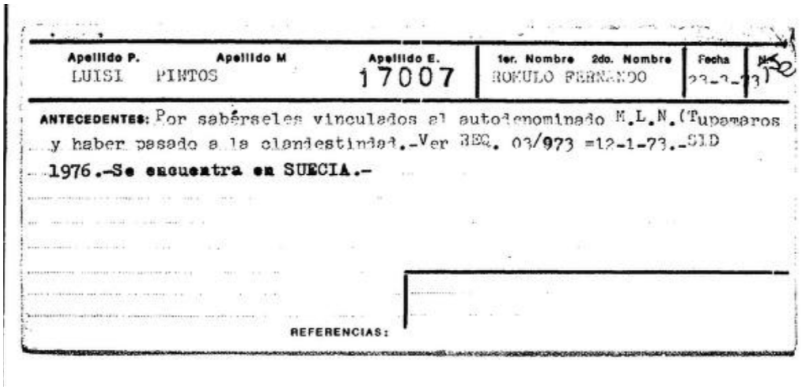


Figura A.2: Ficha de ejemplo

contra una transcripción realizada utilizando Tesseract y sin ningún preprocesamiento adicional. En la imagen de la ficha se puede observar en la figura A.2.

El ejemplo A.2 muestra la transcripción obtenida utilizando Tesseract y sin ningún preprocesamiento adicional de la hoja donde se encuentra la ficha.

Ejemplo A.2: Ejemplo de transcripción obtenida sin preprocesamiento y utilizando el modelo Tesseract para la ficha de la figura A.2.

```
Apellido P. Apellido M Apeliido E. ter. Nombre 2do. Nombre
Fecha 5 LUISI PINTOS i 17007 | roruno rerraxoo  | 27 .2.)3)
y ANTECEDENTES: Por saberseles vinculados 21 autotenominma
do M,L,N .  (Punemeros   - y reber pesado a la elenjestinda
pi REQ, 09/973 =12.1=73,.01D
 1976    -Se encuentra en SUECIA .-
- REFERENCIAS:
```

El ejemplo A.3 muestra la transcripción utilizando el preprocesamiento propuesto en [49], en el cual se segmenta y se alinea la ficha de interés, junto con el modelo MiniCPM-FT-Ext.

Ejemplo A.3: Ejemplo de transcripción obtenida utilizando el preprocesamiento realizado en [49] y el modelo MiniCPM-FT-Ext.

```
Causante: ROLANDO FERNANDO LUISI PINTOS
Nro. Ficha: 17007
Fecha: 23-7-73
Texto: ANTECEDENTES: Por saberseles vinculados al
autodenominado M.L.N. (Tupamaros y haber pasado a la
clandestinidad.- Ver Req.
03/973 = 12-1-73.- SID 1976.- Se encuentra en SUECIA.-
Referencias: None
```

Se observa que la transcripción se encuentra notoriamente más libre de errores a nivel de carácter y de palabras. A su vez, el modelo MiniCPM-FT-Ext logra extraer la transcripción de forma estructurada, clasificando incluso el nombre del causante y el número y fecha de las fichas sin ningún paso adicional de procesamiento, lo cual es una cualidad que es de enorme utilidad para la etapa de extracción de información.

Finalmente, en el ejemplo se logra ver que en la transcripción realizada con Tesseract faltan los nombres de pila del causante, mientras que en la transcripción realizada

con MiniCPM-FT-Ext se extrae el nombre completo, aunque con errores. A pesar de estos errores, lograr transcribir la mayor cantidad de información es de vital importancia para la etapa de extracción de la información, lo cual es otro punto a favor del modelo entrenado.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Ejemplos de instrucciones utilizadas

Dada la extensión de las instrucciones utilizadas, se decidió agregar un ejemplo de cada una en este anexo.

Cada una de las instrucciones se muestra tal cual es recibida por el LLM.

B.1. Instrucción de Código

El ejemplo B.1 muestra una versión completa de la instrucción de código.

Ejemplo B.1: Ejemplo completo de la instrucción de código.

````
Your task is to write Python code that extracts information
    ↪ from a given user text in Spanish.
The following code defines Pydantic classes that represent
    ↪ the information that can be extracted from the text.
Your job is to instatiate the classes with the extracted
    ↪ information.

Here are the Pydantic classes:

```python
from pydantic import BaseModel, Field

class Person(BaseModel):
    """
    A person or group of people joined by some context.
    People can be mentioned by their name, pronouns, titles,
        ↪  aliases or other references.
    """

    span: str = Field(
        ...,
        description='The text of the span.', title='span'
    )
    other_spans: Optional[List[str]] = Field(
````
````

```
        [],
        description='Other spans that mention the entity in
            ↪ the text.',
        title='other_spans',
    )


class Location(BaseModel):
    """
    A physical place or area. Locations can be mentioned by
        ↪ their name, addresses, coordinates or relative
        ↪ references (e.g: 'el local en frente de la
        ↪ comisaria').
    """

    span: str = Field(
        ...,
        description='The text of the span.', title='span'
    )
    other_spans: Optional[List[str]] = Field(
        [],
        description='Other spans that mention the entity in
            ↪ the text.',
        title='other_spans',
    )


class Organization(BaseModel):
    """
    An organization or group of organizations. Organizations
        ↪  can be mentioned by their name, acronyms, aliases
        ↪  or other references.
    """

    span: str = Field(
        ...,
        description='The text of the span.', title='span'
    )
    other_spans: Optional[List[str]] = Field(
        [],
        description='Other spans that mention the entity in
            ↪ the text.',
        title='other_spans',
    )


class Time(BaseModel):
    """
    A time, date or temporal reference. Time values can be
        ↪ mentioned by their date, time of day, relative
        ↪ refrences (e.g: 'ayer') or spans of time (e.g: '
        ↪ del 20 al 30 de mayo').
```

```
    """

    span: str = Field(
        ...,
        description='The text of the span.', title='span'
    )


class Cause(BaseModel):
    """
    A cause as to why someone was detained.
    """

    span: str = Field(
        ...,
        description='The text of the span.', title='span'
    )


class Integrates(BaseModel):
    """
    Specifies that a Person is part of or works for an
        ↪ Organization.
    """

    gov: Person = Field(
        ...,
        description='The Person that is part of or works for
            ↪  the Organization.',
        title='gov',
    )
    dep: Organization = Field(
        ...,
        description='The Organization that the Person
            ↪ integrates.',
        title='dep'
    )


class IsFamily(BaseModel):
    """
    Specifies that two Persons are family-related. This
        ↪ applies to direct family, extended family or
        ↪ inlaws.
    """

    gov: Person = Field(
        ...,
        description='The first Person in the family relation
            ↪ . This is the Person that appears first on the
            ↪  text.',
        title='gov',
```

```
    )
    dep: Person = Field(
        ...,
        description='The second Person in the family
            ↪ relation. This is the Person that appears
            ↪ second on the text.',
        title='dep',
    )


class Contacted(BaseModel):
    """
    Specifies that two Persons have been in contact. It is
        ↪ only used when there's explicit evidence that
        ↪ these two Persons where in contact.
    """

    gov: Person = Field(
        ...,
        description='The first Person in the family relation
            ↪ . This is the Person that appears first on the
            ↪  text.',
        title='gov',
    )
    dep: Person = Field(
        ...,
        description='The second Person in the family
            ↪ relation. This is the Person that appears
            ↪ second on the text.',
        title='dep',
    )


class LivesIn(BaseModel):
    """
    Specifies that a Person lives in a Location.
    """

    gov: Person = Field(
        ...,
        description='The Person that lives in the Location
            ↪ .',
        title='gov'
    )
    dep: Location = Field(
        ...,
        description='The Location where the Person lives.',
        title='dep'
    )


class Detention(BaseModel):
```

```
"""
Detention of a Person or group of Person entities by an
    ↪ authority.
"""

trigger: str = Field(
    ...,
    description='Word or phrase that clearly indicates
        ↪ the occurrence of the event on the text.',
    title='trigger',
)
subject: Optional[List[Person]] = Field(
    None,
    description='The Person or Persons that are detained
        ↪ . If no subjects are mentioned, this field is
        ↪ None.',
    title='subject',
)
src: Optional[Location] = Field(
    None,
    description='The Location where the detention
        ↪ occurred. If no Location source is mentioned,
        ↪ this field is None.',
    title='src',
)
dst: Optional[Location] = Field(
    None,
    description='The Location where the subjects where
        ↪ taken after being detained. If no Location
        ↪ destination is mentioned, this field is None
        ↪ .',
    title='dst',
)
time: Optional[Time] = Field(
    None,
    description='The Time when the event occurred. If no
        ↪  Time is mentioned, this field is None.',
    title='time',
)
cause: Optional[List[Cause]] = Field(
    None,
    description='The list of causes of the detention. If
        ↪  no cause is mentioned, this field is None.',
    title='cause',
)
authority: Optional[Union[Organization, Person]] = Field
    ↪ (
    None,
    description='The Organization or Person that
        ↪ detained the subjects. If no authority is
        ↪ mentioned, this field is None.',
    title='authority',
```

```
    )


class Release(BaseModel):
    """
    Release of a Person or group of Person entities from
        ↪ detention.
    """

    trigger: str = Field(
        ...,
        description='Word or phrase that clearly indicates
            ↪ the occurrence of the event on the text.',
        title='trigger',
    )
    subject: Optional[List[Person]] = Field(
        None,
        description='The Person or Persons that are freed
            ↪ from detention. If no subjects are mentioned,
            ↪ this field is None.',
        title='subject',
    )
    src: Optional[Location] = Field(
        None,
        description='The Location where the subjects were
            ↪ incarcelated and released from. If no Location
            ↪  source is mentioned, this field is None.',
        title='src',
    )
    dst: Optional[Location] = Field(
        None,
        description='The Location were the subjects were
            ↪ taken after being released. This can be their
            ↪ new place of residence or another
            ↪ incarcelation location. If no Location
            ↪ destination is mentioned, this field is None
            ↪ .',
        title='dst',
    )
    time: Optional[Time] = Field(
        None,
        description='The Time when the event occurred. If no
            ↪  Time is mentioned, this field is None.',
        title='time',
    )
    authority: Optional[Union[Organization, Person]] = Field
        ↪ (
        None,
        description='The Organization or Person that decides
            ↪  to release the subjects. If no authority is
            ↪ mentioned, this field is None.',
        title='authority',
```

```
    )
    responsible: Optional[Organization] = Field(
        None,
        description='The Organization that is responsible of
            ↪  monitoring the subjects after their release.
            ↪ If no responsible is mentioned, this field is
            ↪ None.',
        title='responsible',
    )


class Transfer(BaseModel):
    """
    Transfer of a Person or group of Person entities from
        ↪ one location to another. It doesn't include visits
        ↪  or short stays. It can be triggered by forced or
        ↪ voluntary reasons.
    """

    trigger: str = Field(
        ...,
        description='Word or phrase that clearly indicates
            ↪ the occurrence of the event on the text.',
        title='trigger',
    )
    subject: Optional[List[Person]] = Field(
        None,
        description='The Person or Persons that are
            ↪ transferred. If no subjects are mentioned,
            ↪ this field is None.',
        title='subject',
    )
    src: Optional[Location] = Field(
        None,
        description='The Location from where the subjects
            ↪ transferred. If no Location source is
            ↪ mentioned, this field is None.',
        title='src',
    )
    dst: Optional[Location] = Field(
        None,
        description='The Location to were the subjects
            ↪ transferred. If no Location destination is
            ↪ mentioned, this field is None.',
        title='dst',
    )
    time: Optional[Time] = Field(
        None,
        description='The Time when the event occurred. If no
            ↪  Time is mentioned, this field is None.',
        title='time',
    )
```

```
‘‘‘

Here are some examples:

Example ------------------------------------------------
‘‘‘python
# This is the example text
text = """
Causante: Fulano Mengano

Nro. Ficha: None

Fecha: None

Texto: ANTECEDENTES: X/968: se establece que es afiliado a
    ↪ la Asoc. de Estudiantes de Agronom a.- Estando en
    ↪ vigencia el Decreto de M.P.S. fu  detenido por
    ↪ personal de Secc. 20a., por sus expresiones vertidas
    ↪ en un veh culo de transporte colectivo.
Dijo que habr a que derrotar al Gobierno que es
    ↪ estrangulando las libertades de las clases populares y
    ↪  sustituirlo por otro que le diera bienestar al pueblo
    ↪ . 24/1/974:- Detenido por M.P.S. por orden del Senador
    ↪  Jefre de Policia, por figurar en una n mina con
    ↪ otras 12 personas sindicadas por el Ministerio de
    ↪ Educaci n y Cultura como agitadores en Agronom a.-
    ↪ Laura Herrera de El 18/9/74.- APLACADO al P.C. con el
    ↪ 30/9/79.-

Referencias: ARP. C.G. No. 2743
"""

# These are the output instances for this example
entity1 = Person(span='Fulano Mengano', other_spans=[])
entity2 = Organization(span='Asoc. de Estudiantes de
    ↪ Agronom a', other_spans=[])
entity3 = Person(span='personal de Secc. 20a.', other_spans
    ↪ =[])
entity4 = Organization(span='Secc. 20a.', other_spans=[])
entity5 = Organization(span='Gobierno', other_spans=[])
entity6 = Organization(span='otro', other_spans=[])
entity7 = Organization(span='al pueblo', other_spans=[])
entity8 = Person(span='Senador Jefre de Policia',
    ↪ other_spans=[])
entity9 = Organization(span='Policia', other_spans=[])
entity10 = Person(span='otras 12 personas sindicadas por el
    ↪ Ministerio de Educaci n y Cultura', other_spans=['
    ↪ agitadores en Agronom a'])
entity11 = Organization(span='Ministerio de Educaci n y
    ↪ Cultura', other_spans=[])
entity12 = Person(span='Laura Herrera', other_spans=[])
entity13 = Organization(span='P.C.', other_spans=[])
```

```
value1 = Time(span='X/968')
value2 = Cause(span='expresiones vertidas en un veh culo de
    ↪  transporte colectivo')
value3 = Time(span='24/1/974')
value4 = Cause(span='figurar en una n mina con otras 12
    ↪ personas sindicadas por el Ministerio de Educaci n y
    ↪ Cultura como agitadores en Agronom a')
value5 = Time(span='18/9/74')
value6 = Time(span='30/9/79')
relation1 = Integrates(gov=entity1, dep=entity2)
relation2 = Integrates(gov=entity3, dep=entity4)
relation3 = Integrates(gov=entity8, dep=entity9)
relation4 = Integrates(gov=entity10, dep=entity11)
relation5 = Integrates(gov=entity1, dep=entity13)
event1 = Detention(trigger='detenido', subject=[entity1],
    ↪ cause=[value2], authority=entity3)
event2 = Detention(trigger='Detenido', subject=[entity1],
    ↪ time=value3, cause=[value4], authority=entity8)

'''
-----------------------------------------------------


Example -------------------------------------------------
'''python
# This is the example text
text = """
Causante: WALTER ANTONIO BENITEZ REVUELTA

Nro. Ficha: 392

Fecha: None

Texto: ANTECEDENTES: FUE DETENIDO POR G.A.1.* EL DIA
    ↪ 08-12-73.*
CAUSA DE DETENCI N: PERTENECER AL M.L.M.N. (TUPAMAROS).*
SITUACION ACTUAL: DETENIDO EN AVERIGUACIONES:* DE ACUERDO A
    ↪ CIR. RES. No, 7/72.* DEL G.A.1.* DEL DIA 8-12-73 29
    ↪ MAR FA Recinto Brado del H.C.F.F.A.A. El 25-4-1974: (
    ↪ Asociaciones subversivas Atentado a la Constituci n
    ↪ en el grado de conspiraci n seg n los preparatorios
    ↪ y Falsificaci n de documento p blico.-

Referencias: None
"""

# These are the output instances for this example
entity1 = Person(span='WALTER ANTONIO BENITEZ REVUELTA',
    ↪ other_spans=[])
entity2 = Organization(span='G.A.1.', other_spans=['G.A
    ↪ .1.'])
entity3 = Organization(span='M.L.M.N. (TUPAMAROS)',
```

```
    ↪ other_spans=[])
entity4 = Organization(span='H.C.F.F.A.A.', other_spans=[])
value1 = Time(span='08-12-73')
value2 = Cause(span='PERTENECER AL M.L.M.N. (TUPAMAROS)')
value3 = Time(span='8-12-73')
value4 = Time(span='29 MAR')
value5 = Time(span='25-4-1974')
value6 = Cause(span='Asociaciones subversivas')
value7 = Cause(span='Atentado a la Constituci n en el grado
    ↪  de conspiraci n  seg n los preparatorios')
value8 = Cause(span='Falsificaci n de documento p blico')
relation1 = Integrates(gov=entity1, dep=entity3)
event1 = Detention(trigger='DETENIDO', subject=[entity1],
    ↪ time=value1, cause=[value2], authority=entity2)
```
```
---------------------------------------------------------


Example -------------------------------------------------
```python
# This is the example text
text = """
Causante: Maria Esther BENITEZ TONNA

Nro. Ficha: 3591

Fecha: 15-9-73

Texto: ANTECEDENTES: Encuadramiento: Integrante de un Grupo
    ↪ Militar que "funcionaba" en los Montes del Ao.
    ↪ Itacumb , Yutut j  , Cuar  o en los ca averales.-
    ↪ Ultima informaci n obtenida que es detenida el: 23/XI
    ↪ /976.- Movimiento a Subversivo: M.L.N. Ver Oficio No.
    ↪ 250/29/XII/976= Del Estado Mayor Naval= El 18/9/20 se
    ↪ solicita su captura por haberse desviado de la
    ↪ libertad Provisional que gozaba  (Solicitada por el S.
    ↪ T.M.-Req. por Fisco de Rado ECO.- Ref. por Oficio No
    ↪ 5570 al 31/9/80.-
"""

# These are the output instances for this example
entity1 = Person(span='Maria Esther BENITEZ TONNA',
other_spans=[])
entity2 = Organization(span='Grupo Militar',
other_spans=[])
entity3 = Location(span='Montes del Ao. Itacumb ,
    ↪ Yutut j  , Cuar ', other_spans=[])
entity4 = Location(span='los ca averales', other_spans=[])
entity5 = Organization(span='Movimiento a Subversivo: M.L.N
    ↪ .', other_spans=[])
entity6 = Organization(span='Estado Mayor Naval',
    ↪ other_spans=[])
```
```

```
entity7 = Organization(span='S.T.M.', other_spans=[])
entity8 = Organization(span='Fisco de Rado ECO', other_spans
    ↪ =[])
value1 = Time(span='15-9-73')
value2 = Time(span='23/XI/976')
value3 = Time(span='29/XII/976')
value4 = Time(span='18/9/20')
value5 = Time(span='31/9/80')
relation1 = Integrates(gov=entity1, dep=entity2)
relation2 = Integrates(gov=entity1, dep=entity5)
event1 = Detention(trigger='detenida', subject=[entity1],
    ↪ time=value2)
event2 = Release(trigger='libertad Provisional', subject=[
    ↪ entity1])

```

Your output must have the following format:
* You must instantiate one class per line
* You must assign the instance to a variable with the name `
    ↪ entity<index>`, `value<index>`, `event<index>`, or `
    ↪ relation<index>`, where `<index>` is a number starting
    ↪ from 1.
Here's an example of the expected output format:
```python
# Entities
entity1 = Person(<ARGUMENTS>)
entity2 = Location(<ARGUMENTS>)
# Values
value1 = Date(<ARGUMENTS>)
value2 = Cause(<ARGUMENTS>)
value3 = Date(<ARGUMENTS>)
# Relations
relation1 = LivesIn(<ARGUMENTS>)
relation2 = Integrates(<ARGUMENTS>)
# Events
event1 = Detention(<ARGUMENTS>)
```

Your job is to instantiate the classes with the extracted
    ↪ information.
DO NOT modify the given schemas or add any of them.
DO NOT output anything other than the instances of the
    ↪ classes.
DO NOT create any auxiliary functions or classes.
You MUST only output assignment statements that instantiate
    ↪ the classes.
You MUST output valid instances of the schemas.

```python
```

```
# This is the input spanish text
text = """
Causante: ANA MARIA ALVAREZ ANTUNEZ

Nro. Ficha: 1023\n\nFecha: 10/11/72

Texto: ANTECEDENTES: Fue detenida en Paysand  por rapi a
    ↪ de veh culos y lectura de proclama del MLN en
    ↪ colectivo de obreros del Frigor fico CASABLANCA-.\n12
    ↪ /10/71: Fue remitida por: ASOCIACION PARA DELINQUIR,
    ↪ RAPI A , VIOLENCIA PRIVADA y  PRIVACION DE LIBERTAD-.
    ↪ Hab a integrado el grupo de sanidad de la Columna
    ↪ 20/30 Norte, siendo instructora. Tambi n integr  el
    ↪ Sector Militar.=

Referencias: Ver P.E.I. No. 80 YZ 020 RM.3.Dep.2-RM1'
"""

# Instantiate the classes that represent the information
    ↪ extracted
<YOUR CODE HERE>
```

B.2. Instrucción de Lenguaje Natural

El ejemplo B.2 muestra una versión completa de la instrucción de lenguaje natural.

Ejemplo B.2: Ejemplo completo de la instrucción de lenguaje natural.

```
Your task is to write natural language statements that
    ↪ extract information from a given text in Spanish.
The following section defines the information that must be
    ↪ extracted from the text and rules for the output
    ↪ format.

Here are the definitions and rules:

Rules --------------------------------
Entities:
    - Person: A person or group of people joined by some
        ↪ context. People
    can be mentioned by their name, pronouns, titles,
        ↪ aliases or other
    references. A statement that identifies an entity of
        ↪ this class must
    have the following format: <ENTITY_NAME> is a Person.
    - Location: A physical place or area. Locations can be
        ↪ mentioned by
    their name, addresses, coordinates or relative
        ↪ references (e.g:
    'el local en frente de la comisaria'). A statement that
        ↪ identifies
```

```
    an entity of this class must have the following format:
        ↪ <ENTITY_NAME>
    is a Location.
    - Organization: An organization or group of
        ↪ organizations. Organizations
    can be mentioned by their name, acronyms, aliases or
        ↪ other references.
    A statement that identifies an entity of this class must
        ↪  have the
    following format: <ENTITY_NAME> is a Organization.
Values:
    - Time: A time, date or temporal reference. Time values
        ↪ can be
    mentioned by their date, time of day, relative refrences
        ↪  (e.g: 'ayer')
    or spans of time (e.g: 'del 20 al 30 de mayo'). A
        ↪ statement that
    identifies a value of this class must have the following
        ↪  format:
    <VALUE> is a Time.
    - Cause: A cause as to why someone was detained. A
        ↪ statement that
    identifies a value of this class must have the following
        ↪  format:
    <VALUE> is a Cause.
Relations:
    - Integrates: Specifies that a Person is part of or
        ↪ works for an
    Organization.. This relation connects an entity of the
        ↪ class Person
    to an entity of the class Organization. A statement that
        ↪  identifies
    a relation of this class must have the following format:
        ↪  Person
    <ENTITY_NAME> integrates Organization <ENTITY_NAME>.
    - IsFamily: Specifies that two Persons are family-
        ↪ related. This
    applies to direct family, extended family or inlaws..
        ↪ This relation
    connects an entity of the class Person to an entity of
        ↪ the class
    Person. A statement that identifies a relation of this
        ↪ class must
    have the following format: Person <ENTITY_NAME> is
        ↪ family of Person
    <ENTITY_NAME>.
    - Contacted: Specifies that two Persons have been in
        ↪ contact. It is
    only used when there's explicit evidence that these two
        ↪ Persons where
    in contact.. This relation connects an entity of the
        ↪ class Person to
```

```
    an entity of the class Person. A statement that
        ↪ identifies a relation
    of this class must have the following format: Person <
        ↪ ENTITY_NAME>
    contacted Person <ENTITY_NAME>.
    - LivesIn: Specifies that a Person lives in a Location..
        ↪  This
    relation connects an entity of the class Person to an
        ↪ entity of the
    class Location. A statement that identifies a relation
        ↪ of this class
    must have the following format: Person <ENTITY_NAME>
        ↪ lives in
    Location <ENTITY_NAME>.
Events:
    - For ALL events: if an event argument is unknown, you
        ↪ must complete
    the <ENTITY_NAME> or <VALUE> with <UNKNOWN>.
    - ALL event statements must have a trigger identified at
        ↪  the end of
    the statement. This is the only mandatory field.
    - Detention: Detention of a Person or group of Person
        ↪ entities by an
    authority. This event has the following arguments:
            * trigger: this field is a string. Word or
                ↪ phrase that
            clearly indicates the occurrence of the event on
                ↪  the text.
            * Subject: this field is a comma-separated list
                ↪ of Person
            entities. The Person or Persons that are
                ↪ detained.
            * Authority: this field is an Organization or
                ↪ Person entity.
            The Organization or Person that detained the
                ↪ subjects.
            * Time: this field is a Time entity. The time at
                ↪  which the
            detention occurred.
            * Cause: this field is a comma-separated list of
                ↪  Cause
            entities. The reasons for the detention.
            * Source: this field is a Location entity. The
                ↪ Location
            where the detention occurred.
            * Destination: this field is a Location entity.
                ↪ The Location
            where the detained subjects were taken.
        A statement that identifies an event of this class
            ↪ must have the
        following format: Person <ENTITY_NAME> (or a comma-
            ↪ separated list
```

```
of Person entities) was (or 'were' if there's more
    ↪ than one
person) detained by Organization or Person <
    ↪ ENTITY_NAME> at Time
<VALUE> and Location <ENTITY_NAME> because of Cause
    ↪ <VALUE> (or
a comma-separated list of Cause), and was (or 'were
    ↪ ') tranferred
to Location <ENTITY_NAME>. The trigger is <TRIGGER>.
- Release: Release of a Person or group of Person
    ↪ entities from
detention. This event has the following arguments:
    * trigger: this field is a string. Word or
        ↪ phrase that
    clearly indicates the occurrence of the event on
        ↪  the text.
    * Subject: this field is a comma-separated list
        ↪ of Person
    entities. The Person or Persons that are freed
        ↪ from
    detention.
    * Authority: this field is an Organization or
        ↪ Person entity.
    The Organization or Person that decides to
        ↪ release the
    subjects.
    * Time: this field is a Time entity. The time at
        ↪  which the
    release occurred.
    * Source: this field is a Location entity. The
        ↪ Location
    where the subjects were incarcelated and
        ↪ released from.
    * Destination: this field is a Location entity.
        ↪ The
    Location where the subjects were taken after
        ↪ being released.
    * Responsible: this field is a Organization
        ↪ entity. The
    Organization that is responsible of monitoring
        ↪ the subjects
    after their release.
A statement that identifies an event of this class
    ↪ must have the
following format: Person <ENTITY_NAME> (or a comma-
    ↪ separated
list of Person entities) was (or 'were' if there's
    ↪ more than one
person) released by Organization or Person <
    ↪ ENTITY_NAME> at Time
<VALUE> and Location <ENTITY_NAME>, and was (or '
    ↪ were')
```

```
        tranferred to Location <ENTITY_NAME>, under the
            ↪ responsibility
        of Organization <ENTITY_NAME>. The trigger is <
            ↪ TRIGGER>.
        - Transfer: Transfer of a Person or group of Person
            ↪ entities
        from one location to another. It doesn't include
            ↪ visits or short
        stays. It can be triggered by forced or voluntary
            ↪ reasons. This
        event has the following arguments:
            * trigger: this field is a string. Word or
                ↪ phrase that
            clearly indicates the occurrence of the event on
                ↪  the text.
            * Subject: this field is a comma-separated list
                ↪ of Person
            entities. The Person or Persons that are
                ↪ transferred.
            * Source: this field is a Location entity. The
                ↪ Location from
            where the subjects transferred.
            * Destination: this field is a Location entity.
                ↪ The Location
            to were the subjects transferred.
            * Time: this field is a Time entity. The time at
                ↪  which the
            transfer occurred.
        A statement that identifies an event of this class
            ↪ must have the
        following format: Person <ENTITY_NAME> (or a comma-
            ↪ separated
        list of Person entities) was (or 'were' if there's
            ↪ more than one
        person) transferred to Location <ENTITY_NAME> at
            ↪ Time <VALUE>
        from Location <ENTITY_NAME>. The trigger is <TRIGGER
            ↪ >.
 End of the rules -----------------------------------

Here are some examples:
Example ------------------------------------------------
-> This is the example text:
Causante: JOSE RODRIGUEZ

Nro. Ficha: 1920

Fecha: None

Texto: ANTECEDENTES: Integrante del R.O.E., reclutado en
    ↪ mediados de
noviembre, 1972 por la R  l  PEREZ al infarto, realiz  un
```

```
    ↪ atentado
contra la del sec- fio Juan GOMEZ en la Ciudad de La Plata
(Funcionario de la F.M.E.) con cocktel es molotov en la
    ↪ cuadra
particaron adem s  Mar a del Carmen Neri y Alicia Lopez "
    ↪ Canosa"
Jorge Lemes, Juana Lemes y explosivos fueron confeccionados
    ↪ seg n el
depo- nimo Coghl n.-

Referencias: None
-> These are the statements:
List of entities:
JOSE RODRIGUEZ is a Person
R.O.E. is an Organization
R l PEREZ is a Person
Juan GOMEZ is a Person
Ciudad de La Plata is a Location
F.M.E. is an Organization
Mar a del Carmen Neri is a Person
Mar a del Carmen Neri y Alicia Lopez "Canosa" Jorge Lemes,
    ↪ Juana
Lemes is a Person
Alicia Lopez "Canosa" is a Person
Jorge Lemes is a Person
Juana Lemes is a Person
List of values:
mediados de noviembre, 1972 is a Time
realiz  un atentado contra la del sec- fio Juan GOMEZ en la
    ↪  Ciudad
de La Plata (Funcionario de la F.M.E.) con cocktel es
    ↪ molotov is a
Cause
List of relations:
Person JOSE RODRIGUEZ integrates Organization R.O.E.
Person JOSE RODRIGUEZ contacted Person R l PEREZ
Person Juan GOMEZ integrates Organization F.M.E.
Person JOSE RODRIGUEZ contacted Person Mar a del Carmona
    ↪ Neri y
Alicia Lopez "Canosa" Jorge Lemes, Juana Lemes
List of events:

End of example ------------------------------------------
Example -------------------------------------------------
-> This is the example text:
Causante: Fulano Mengano

Nro. Ficha: 3507

Fecha: None

Texto: ANTECEDENTES://... mismo; donde permanecieron cuatro
```

```
    ↪ meses,
recibiendo instrucci n militar, de tiro, marchas y cursos
    ↪ de
adiestramiento.- 20) que luego Policia le incaut  una
    ↪ pistola
Luger perteneciente al movimiento.- Alfredo MUSSIO declar
    ↪ en
Acta que el causante junto con Nebio MELO y Arturo
    ↪ ETCHENIQUE
fueon en su compa  a en 1967 quedar n all  el causante.-
    ↪  Particip
en las acciones (P.E.I.70/72Anxo 9)- Asalto a una armamer a
    ↪ , roba
50 armas, junto con GABINO FALERO, El Tama, CATALANO y 2
    ↪ mujeres.-
Asalto a un Banco en Avda. Itsalia junto con Jos  DENAT
    ↪ BAXTER,
Jos  Luiz NELL TACCHI, RODRIGUEZ IRIMOIN, "El Pata" y

Referencias: None
-> These are the statements:
List of entities:
Fulano Mengano is a Person
Policia is an Organization
movimiento is an Organization
Alfredo MUSSIO is a Person
Nebio MELO is a Person
Nebio MELO y Arturo ETCHENIQUE is a Person
Arturo ETCHENIQUE is a Person
all  is a Location
GABINO FALERO is a Person
GABINO FALERO, El Tama, CATALANO y 2 mujeres is a Person
El Tama is a Person
CATALANO is a Person
2 mujeres is a Person
Banco is an Organization
Avda. Itsalia is a Location
Jos  DENAT  BAXTER is a Person
Jos  DENAT  BAXTER, Jos  Luiz NELL TACCHI, RODRIGUEZ
    ↪ IRIMOIN,
"El Pata" is a Person
Jos  Luiz NELL TACCHI is a Person
RODRIGUEZ IRIMOIN is a Person
"El Pata" is a Person
List of values:
cuatro meses is a Time
luego is a Time
1967 is a Time
Asalto a una armamer a, roba 50 armas is a Cause
Asalto a un Banco en Avda. Itsalia is a Cause
List of relations:
Person Alfredo MUSSIO contacted Person Fulano Mengano
```

```
Person Fulano Mengano contacted Person Nebio MELO y Arturo
ETCHENIQUE
Person Fulano Mengano contacted Person GABINO FALERO, El
    ↪ Tama,
CATALANO y 2 mujeres
Person Fulano Mengano contacted Person Jos  DENAT  BAXTER,
Jos  Luiz NELL TACCHI, RODRIGUEZ IRIMOIN, "El Pata"
List of events:

End of example ------------------------------------------
Example -------------------------------------------------
-> This is the example text:
Causante: SERGIO DANIEL FERNANDEZ CANEPA

Nro. Ficha: 10863

Fecha: None

Texto: ANTECEDENTES: Fu  detenido por Bn. Ing. 1 por
    ↪ presuntamente
colaboraci n con el M.L.N.; posteriormente fue detenido el
hermano: Jos  H ber FERNANDEZ CANEPA, el cual estaba
    ↪ requerido por
la misma causa.- Ver Parte de Opera CION No. 7497.* del Bn.
    ↪ Ing. 1
del d a 3-3-74.* Ver Informe de la OFICINA S-2 del Bn. Ing.
    ↪  1 del
08-MAR-974.-

Referencias: None
-> These are the statements:
List of entities:
SERGIO DANIEL FERNANDEZ CANEPA is a Person
Bn. Ing. 1 is an Organization
M.L.N. is an Organization
el hermano: Jos  H ber FERNANDEZ CANEPA is a Person
OFICINA S-2 del Bn. Ing. 1 is an Organization
List of values:
presuntamente colaboraci n con el M.L.N is a Cause
3-3-74 is a Time
08-MAR-974 is a Time
List of relations:
Person SERGIO DANIEL FERNANDEZ CANEPA is family of Person el
hermano: Jos  H ber FERNANDEZ CANEPA
List of events:
Person SERGIO DANIEL FERNANDEZ CANEPA was detained by
    ↪ Organization
Bn. Ing. 1 at <UNKNOWN> and <UNKNOWN> because of Cause
    ↪ presuntamente
colaboraci n con el M.L.N, and was transferred to <UNKNOWN
    ↪ >. The
trigger is detenido.
```

```
Person el hermano: Jos  H ber FERNANDEZ CANEPA was
    ↪ detained by
<UNKNOWN> at <UNKNOWN> and <UNKNOWN> because of Cause
    ↪ presuntamente
colaboraci n con el M.L.N, and was transferred to <UNKNOWN
    ↪ >. The
trigger is detenido.

End of example ----------------------------------------
End of the example sections

Your output must have the following format:
* You must provide one statement per line
* You must first define all entities, then all values, then
    ↪ all
relations, and finally all events, following the rules
    ↪ provided
before.
* If an event argument is unknown, you must complete the
<ENTITY_NAME> or <VALUE> with <UNKNOWN>

Here's an example of the expected output:

Example ----------------------------------
List of entities:
<ENTITY_NAME> is a Person
<ENTITY_NAME> is a Location
<ENTITY_NAME> is an Organization
List of values:
<VALUE> is a Time
<VALUE> is a Cause
<VALUE> is a Time
List of relations:
Person <ENTITY_NAME> lives in Location <ENTITY_NAME>
Person <ENTITY_NAME> integrates Organization <ENTITY_NAME>
List of events:
Person <ENTITY_NAME>, Person <ENTITY_NAME> were detained by
Organization <ENTITY_NAME> at Time <VALUE> and Location
<ENTITY_NAME> because of Cause <VALUE>, and were tranferred
    ↪ to
Location <ENTITY_NAME>. The trigger is <TRIGGER>.
Person <ENTITY_NAME> was transferred to Location <
    ↪ ENTITY_NAME> at
Time <VALUE> from Location <ENTITY_NAME>. The trigger is <
    ↪ TRIGGER>.
End of example ----------------------------------------

Your job is to provide statements that extract the
    ↪ information
from the text.
DO NOT modify the given rules or add any of them.
DO NOT output anything other than the statements on the
```

```
    ↪ specified
format.
DO NOT create any auxiliary rules.
You MUST only output valid statements.
Events MUST STRICTLY FOLLOW the format mentioned on the
    ↪ examples,
with the same order of arguments and without changing the
structure.
DO NOT extract information from the given examples.

Input text:

Causante: JOSE EDUARDO LEMOS INSUA

Nro. Ficha: 16285

Fecha: 14 Ago. 73

Texto: ANTECEDENTES: Particip  en el atentado contra el
    ↪ domicilio
del se or M ndez (Funcionario de A.F.E.), arrojado
    ↪ cockteles
molotov junto a Jos  Michelini y Pablo Herrandonea mientras
    ↪  hac an
de "Campana" sus compa eros Alicia Verde y Mar a del
    ↪ Carmen Vidal.
-Fecha de detenci n 14.8.73, por Bn.Ing.1.-

Referencias: None

Output statements:
```

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Análisis de los embeddings de los datos de entrenamiento

Este anexo presenta un breve análisis de los embeddings obtenidos para los textos del conjunto de datos de entrenamiento durante la construcción del módulo de Recuperación de Ejemplos. Los embeddings fueron obtenidos utilizando el modelo BGE-M3 [12].

C.1. Análisis

La figura C.1 muestra una representación en dos dimensiones de los embeddings. La reducción de dimensiones de los embeddings fue obtenida utilizando el algoritmo *Principal Component Analysis* (PCA). A su vez, se agrupan los embeddings reducidos en 5 clusters utilizando el algoritmo K-Means.

Incluso luego de reducidas las dimensiones, se observa que los embeddings mantienen un ordenamiento espacial que permite agruparlos. Esto es esperable, ya que los textos del conjunto de entrenamiento presentan muchas similitudes y naturalmente se dan grupos de textos que hablan sobre temas parecidos o hacen menciones a eventos o datos similares. Por esto, es importante notar que los embeddings capturan al menos una noción de grupos.

A continuación, se visualizan algunos ejemplos de textos de cada grupo para analizar si los mismos tienen coherencia.

El ejemplo C.1 muestra tres textos seleccionados aleatoriamente del Cluster 1. Estos tres textos parecen no tener una similitud significativa. Puede deberse a que el Cluster 1 es demasiado disperso, y que la reducción de dimensionalidad haya eliminado alguna de las dimensiones que diferencian a estos textos.

Ejemplo C.1: Textos de ejemplo seleccionados aleatoriamente del Cluster 1.

```
ANTECEDENTES: 13/7/70: EL POPULAR: -Integrante del nuevo
    ↪ Consejo Administrativo de la "UNION AUTONOMA DE
    ↪ OBREROS Y EMPLEADOS DEL GAS Y DIQUE MAUA, en calidad
    ↪ de suplente para el a o 1970.-

ANTECEDENTES: Su nombre y direcci n aparecieron en una
    ↪ liberta utada en la AGENCIA SINJUA (CHINA POPULAR), en
    ↪  los 1ros, de  julio del/72, cuando se procedi  a
```

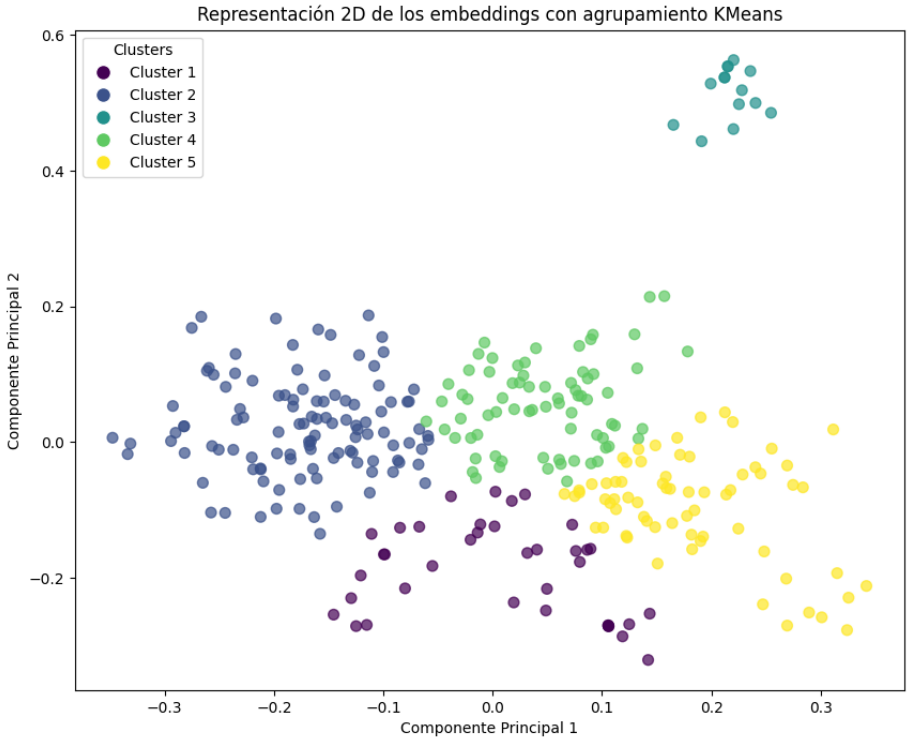


Figura C.1: Representación 2D obtenida con PCA de los embeddings de los textos del conjunto de datos de entrenamiento. Los embeddings reducidos fueron agrupados utilizando el algoritmo K-Means en 5 clusters.

```
    ↪ retirar un caj n con rollos de pel cula por parte de
    ↪  OCOA.-

ANTECEDENTES: - Su nombre apareci  en una relaci n de
    ↪ personas pertenecientes al parido socialista a las que
    ↪  el Arq. Ram n MARTINEZ GUARINO, les efectuaba la
    ↪ cobranza por Oficio 40/I/72 del AAA1, del 17-may-72.
```

El ejemplo C.2 muestra tres textos seleccionados aleatoriamente del Cluster 2. Estos textos presentan una similitud interesante: todos hablan de detenciones. El Cluster 2 puede estar agrupando textos que mencionen las palabras "detenido" o "procesado", lo cual puede ser muy útil a la hora de recuperar ejemplos, ya que se le pueden dar al LLM ejemplos específicos de cómo extraer eventos del tipo `Detención`.

Ejemplo C.2: Textos de ejemplo seleccionados aleatoriamente del Cluster 2.

```
ANTECEDENTES DETENIDO e INTERNADO POR VIOLAT LAS MPS EN 1969
    ↪  EN G.M.A.4.-

ANTECEDENTES: 7/5/72 Es detenido en Millan y jaba en una
    ↪ camioneta Matr. 112.097. con los sediciosos Leonel
    ↪ Javier HARARI DUBINSKY, Nelida Raquel MAZAS DOTTI,
```

```
    ↪ Carlos Eugenio FURTDADO TOPOLANSKY, Enrique TURELL
    ↪ TAGLIABUE, Jose Martiniano ZAPATA ACUMA.- Ver
    ↪ Memorandum No. 47 del DPT.3 de la DNII del 19/2/73.-
    ↪ Liberado el 12-9-73 Art. 168 CF.

ANTECEDENTES: Detenido. Bn. Ing. No. 4.- Bn. 332.- La Barra.
```

El ejemplo C.3 muestra tres textos seleccionados aleatoriamente del Cluster 3. Los textos de este grupo tienen una similitud claramente visible. Todos están escritos siguiendo una estructura similar y utilizan exactamente las mismas palabras. Se puede observar en la figura C.1 cómo este grupo es el más alejado y el más fácil de separar del resto. Esto evidencia la presencia de múltiples textos muy similares en los datos de entrenamiento, lo cual puede tener efectos contraproducentes si los mismos se utilizaran para entrenar o realizar fine-tuning.

Ejemplo C.3: Textos de ejemplo seleccionados aleatoriamente del Cluster 3.

```
ANTECEDENTES: Argentino, documento CIPF 6.558.170.- El mismo
    ↪  es terrorista argentino, seg n parte especial de
    ↪ informaci n Nro. 163/BE/81 del Dpto. II/EME del
    ↪ 6-4-81

ANTECEDENTES:
El mismo es terrorista argentino, seg n Parte Especial de
    ↪ Informaci n No. 163/BE/981 del Dpto. 2 del E.M.E. del
    ↪  6-4-981.-

ANTECEDENTES:
La misma es terrorista argentina, seg n Parte Especial del
    ↪ 163/BE/981 del Dpto. 2 del E.M.E. del 6-4-981.-
```

El ejemplo C.4 muestra tres textos seleccionados aleatoriamente del Cluster 4. Este grupo, al igual que el Cluster 1, no presenta ninguna similitud significativa. Aplican los mismos comentarios ya expresados para el Cluster 1.

Ejemplo C.4: Textos de ejemplo seleccionados aleatoriamente del Cluster 4.

```
ANTECEDENTES: Seg n Memo. Nro. 2339/981 del 5-jun-81 de la
    ↪ DNII
a sol. Nro. 024/81 anotado en 15-mayo-81 Registro: LOPEZ
    ↪ DUBINI
LUIS EDGARDO, Co. Nro. 970.638 domicilio Juan Carlos Blanco
    ↪ 2
(en 1974) Occupaci n, bancario (BROU) nacio el d a 5-3944,
    ↪  hijo
de Elbio y de Carolina, casado.-
AMOTACIONICN: 5-8-968, Estando en vigencia el decreto sobre
M.P.S. fue detenido en averiguaci n, se trata de un
    ↪ funcionario
de la Secci n Jur dica del BROU, activo dirigente gremial,
siendo interno en la Escuela de Armas y Servicios, a
    ↪ disposici n
del Comando de la Regi n Militar Nor.1.-
25-2-969: Seg n informe Nro. 246 partic p  en la reuni n
    ↪  de
```

```
bancarios////

ANTECEDENTES: Procesada por el delito de "Asociaci n para
delinquir".-

ANTECEDENTES: VER P.E.I. 341/73 del d a 250800ct973.-
```

El ejemplo C.5 muestra tres textos seleccionados aleatoriamente del Cluster 5. Este grupo también presenta una similitud interesante: los textos hablan de personas que integran listas, comisiones, organizaciones o comités. Puede ser de utilidad para recuperar ejemplos que expliquen al LLM cómo extraer entidades de tipo `Organización` o relaciones de tipo `Integra` o `Contacto`.

Ejemplo C.5: Textos de ejemplo seleccionados aleatoriamente del Cluster 5.

```
10/09/82 Figura con el No 77 como candidato por el Partido
    ↪ Nacional identificado con las letras EAM pertenece a
    ↪ la  agrupaci n "POR LA UNION NACIONALISTA Y EL
    ↪ PARTIDO NACIONALISTA HERRERISMO" a las elecciones
    ↪ internas del 28/11/82 (Seg n solicitud de registro de
    ↪  dicha lista de candidatura ante la Junta Electoral de
    ↪  Montevideo)

ANTECEDENTES: Columna 24 zona de Paysand .- Integr  la
    ↪ Organizaci n por su gran amistad con el Sedicioso
    ↪ RAUL SENDIC desde los comienzos de esta.- Ocult  a
    ↪ SENDIC en diversas oportunidades cuando este realizaba
    ↪  tareas para los Departamentos de: Rivera y Artigas.-
    ↪ Cuando Sendic estuvo enterrado en las inmediaciones de
    ↪  la Ciudad de Paysand  BLANCO era el encargado de
    ↪ proporcionarle alimentaci n.- A lo  ltimo  se le
    ↪ hab a ordenado construir un berret n en una
    ↪ carpinter a de su propiedad hecho que se neg  a
    ↪ llevarlo a cabo.-

14-6-83.- El 29-5-83 el mismo es elegido para integrar el
    ↪ Comit  Ejecutivo Departamental del Partido Colorado
    ↪ por la lista 15 sub-lemma "UNIDAD Y REFORMA". (Seg. P.
    ↪  de Nov. 29 de la D.I.I. del 29-5-83).-
```

C.2. Conclusiones

Este análisis superficial de los embeddings obtenidos revela que estos logran capturar información relevante a la hora de agrupar o buscar textos por similitud semántica. Con una reducción de dimensiones bastante agresiva, se preservan grupos que hablan sobre los mismos tipos de eventos, relaciones o temáticas.

Esto es de suma importancia para el enfoque tomado en el diseño del sistema. Al querer extraer información de un nuevo texto, estos embeddings permiten buscar ejemplos similares que se utilizan para dar mejores instrucciones al LLM sobre cómo extraer la información y generar la salida.

Apéndice D

Resolución de Correferencias

La tarea de resolución de correferencias trata de identificar en el texto menciones a entidades (objetos físicos u eventos) y agrupar todas aquellas menciones correferentes a una misma entidad [27]. Por ejemplo, en la siguiente porción de texto:

```
A pesar de que Martín y su perro disfrutan de salir a pasear,
a veces le gusta más jugar al Nintendo.
```

Para un humano puede ser sencillo determinar que a quien le gusta más jugar al Nintendo es a Martín y no a su perro, pero para un programa de computadora esto no es tan directo. En este caso, el problema de resolver a qué entidad refiere la mención "le" (si a la entidad perro o a la entidad Martín) requiere de conocimiento sintáctico pero a su vez semántico, ya que es muy improbable que el perro de Martín juegue al Nintendo. Si cambiamos la oración por una sintácticamente parecida, como:

```
A pesar de que Martín y su perro disfrutan de salir a pasear, a veces
le gusta más jugar con las sobras de huesos.
```

Podemos decir que, con mayor probabilidad, la entidad asociada a la mención "le" en este caso sea el perro, puramente por conocimiento semántico.

Esta tarea tiene una gran importancia en el área del procesamiento del lenguaje natural, en particular por su utilidad para resolver otras tareas de mayor nivel como reconocimiento de entidades nombradas (NER, por su nombre en inglés *Named Entity Recognition*), contestar preguntas (QA, por su nombre en inglés *Question Answering*), enlazado de entidades (*Entity Linking*) o extracción de información (IE, por su nombre en inglés *Information Extraction*), donde es importante que los modelos produzcan respuestas coherentes, informadas y contextualmente acertadas.

El resto de este capítulo presenta un corto resumen del estado del arte para la tarea de resolución de correferencias. Esta revisión del estado actual de la tarea fue vital para estudiar la viabilidad de una solución para los documentos del pasado reciente.

D.1. Métricas

Esta sección describe las métricas comúnmente utilizadas para evaluar la calidad de los sistemas de resolución de correferencias. Al hablar de estas métricas, se suelen utilizar los términos *key* y *response clusters*, para referirse a las entidades (referidas como *clusters*) presentes en el conjunto etiquetado y las obtenidas por el sistema respectivamente.

D.1.1. Mention Matching

Previo a calcular la eficiencia del sistema en cuanto a los enlaces de correferencia y las entidades construidas, se tiene que determinar la correspondencia entre las menciones presentes en las entidades *key* y las entidades *response*.

Para entender la necesidad de este paso, consideremos el siguiente ejemplo. Supongamos que tenemos el siguiente fragmento de texto con las entidades *key* marcadas con corchetes rectos:

```
La [1 escuela número 52 ] inauguró este [5 miércoles ] su [2 nueva cancha
de fútbol ]. [1 La institución ] ahora cuenta con [3 tres canchas ] que
serán utilizadas en [4 los campeonatos departamentales ].
```

Un sistema de resolución de correferencias puede crear la siguiente respuesta:

```
[1 La escuela ] número 52 inauguró este miércoles su nueva [2 cancha
de fútbol ]. [1 La institución ] ahora cuenta con [3 tres canchas ] que
serán utilizadas en [4 los campeonatos departamentales ].
```

No es directo para un programa realizar la asociación entre menciones en la respuesta y menciones en la *key*. Por ejemplo, ¿cómo se determina que la mención "La escuela" está asociada a la mención "escuela número 52"? ¿o qué no hay ninguna mención asociada a la mención "miércoles"? Determinar dicha asignación es imprescindible para luego calcular la calidad de los enlaces de correferencia o los *clusters* formados. Por ende, en [91] se plantean diversas estrategias para hacer esta asignación.

Exact Matching

Esta es la asignación más sencilla, que determina que dos menciones están asociadas si ambas contienen exactamente las mismas palabras en la misma posición del texto.

Bajo esta asignación, en el ejemplo anterior, las menciones "nueva cancha de fútbol" y "cancha de fútbol" no estarían asociadas, a pesar de contener la información necesaria para referir a la misma entidad.

Al ser este enfoque demasiado estricto, se desarrollaron otros enfoques más laxos y que contemplan diferencias mínimas en la detección de menciones.

Partial Matching

En este caso, dada una mención *key* y una mención *response*, se considera que están asociadas si la mención *key* contiene todas las palabras de la mención *response* (es decir, la mención *response* no tiene palabras que no están en la *key*) y a su vez, la "cabeza" de la mención *key* está incluida en la mención *response*.

La cabeza de una mención es la parte central de la mención, que usualmente aporta la mayor información sobra la entidad a la que refiere esa mención. Es un componente sintáctico, aunque suele estar cargado de significado semántico. Por ejemplo, en la mención "nueva cancha de fútbol", la cabeza sería la palabra "cancha".

En el ejemplo anterior, las menciones "La escuela" y "escuela número 52" no estarían asociadas, ya que la primera contiene una palabra que no está en la mención *key*. Las menciones "cancha de fútbol" y "nueva cancha de fútbol" si estarían asociadas, ya que la primera está incluida en la segunda, y la primera contiene la cabeza de la mención ("cancha").

En varios casos reales, se pueden tener menciones incluidas dentro de otras, lo cual puede llevar a que el criterio anterior sea ambiguo y una mención *key* o *response* pueda tener más de una asociación. Por lo tanto, también se aplican las siguientes reglas de

desambiguación. Dada una mención m de un documento, que puede asociarse a más de una mención n de otro documento:

1. elegir la mención que se superpone con m en mayor medida
2. Si aún queda más de una mención n, elegir la que comienza antes en el texto
3. Si aún queda más de una mención n, elegir la que comienza después en el texto

Para ver un caso donde son necesarias estas reglas, supongamos que el sistema retorna dos menciones "cancha de" y "cancha de fútbol". Ambas se podrían asociar a la mención *key* "nueva cancha de fútbol" utilizando el criterio previamente descrito, pero aplicando las reglas de desambiguación, se elige la mención "cancha de fútbol" como la que se debería asociar.

Algo importante a notar de este enfoque para la asociación de menciones es que requiere de la detección de las palabras cabeza dentro de las menciones. Este paso a ser un requerimiento necesario de los sistemas de resolución de correferencia de las conferencias CRAC 2022 [91] y CRAC 2023 [90]. Se observa que la cabeza de una mención se puede definir sintácticamente observando el árbol de dependencias sintácticas de la oración, y que esta información suele estar disponible en los conjuntos etiquetados de correferencias y suele ser aceptado su uso en los sistemas evaluados en las conferencias.

Head Matching

En la conferencia CRAC 2022 [91] se utilizó Partial Matching como métrica de evaluación de los sistemas. Esto llevó a que, con el fin de optimizar dicha métrica, muchos de los sistemas presentados en esa conferencia atinaran a detectar menciones como solo la cabeza de la mención y no la mención completa. Esta estrategia reduce la cantidad de menciones detectadas sin asociación, pero no es tan útil en un caso de uso real como detectar menciones completas o de mayor cobertura.

A razón de esto, en la conferencia CRAC 2023 [90] se presentó y utilizó el método *Head Matching* para asociar menciones *key* y *response*. Este método considera que dos menciones están asociadas si las cabezas indicadas para ambas contienen exactamente los mismos *tokens*. Si hay más de una mención que cumpla con este criterio, entonces se considera la mención completa y se aplican las mismas reglas de desambiguación que para Partial Matching.

Se observa que esta asociación es más laxa que la de Partial Matching, y por ende no penaliza a sistemas que detecten menciones que incluyen palabras que no deberían, siempre que la cabeza detectada coincida con la de la *key*. Este criterio tiene el objetivo de que los sistemas que predicen menciones completas, y por ende se ajustan mejor a casos de uso reales, puedan obtener resultados justos y comparables con los demás sistemas, siempre y cuando detecten correctamente las cabezas de las menciones.

D.1.2. Detección de Menciones

Los métodos descritos en la anterior sección son útiles para luego calcular la calidad de los enlaces de correferencia, pero su resultado final es una asociación entre menciones y no una medida de calidad de las menciones detectadas. Por eso, en [91] se introduce la métrica MOR (*Mention Overlap Ratio*), como una medida de calidad de las menciones detectadas, independientemente de a que entidad hayan sido asignadas.

Dados los conjuntos K y R de todas las menciones *key* y *response* respectivamente, el primer paso es hallar la asignación $A(K, R)$ que maximiza la cantidad de palabras superpuestas entre menciones *key* y *response*. Notesé que esta asignación es distinta

de las descriptas previamente e ignora las cabezas de las menciones. Luego, el Recall de MOR se calcula como:

$$MOR_{rec} = \frac{\sum_{(k,r)\in A(K,R)} |k \cap r|}{\sum_{k\in K} |k|}$$

Se puede interpretar como la razón entre la cantidad de palabras superpuestas y la cantidad total de palabras en las menciones *key*. Analogamente, la Precision de MOR se calcula como:

$$MOR_{prec} = \frac{\sum_{(k,r)\in A(K,R)} |k \cap r|}{\sum_{r\in R} |r|}$$

Que se puede interpretar como la razón entre la cantidad de palabras superpuestas y la cantidad total de palabras en las menciones *response*.

D.1.3. Enlaces de Correferencia

Las métricas descriptas a continuación asumen que los *key* y *response clusters* contienen las mismas menciones, y son utilizadas para evaluar únicamente la conformación de las entidades o los enlaces de correferencia. Estas métricas no se preocupan por que las menciones detectadas sean las mismas o que estas estén alineadas, aunque el hecho de que hayan menciones faltantes o sobrantes en las entidades *response* seguramente afecte las métricas de enlaces de correferencias.

Denominaremos K y R los conjuntos de entidades o *clusters* en la *key* y en la *response* respectivamente. Las siguientes descripciones de las métricas fueron extraídas del trabajo [45], que evalúa diversas de estas métricas y propone una nueva.

MUC

MUC [70] es una de las primeras métricas utilizadas para evaluar sistemas de correferencia. Es una métrica *link-based* o fuertemente dependiente de los enlaces de correferencia. El Recall de MUC se puede calcular como:

$$\text{Recall } = \frac{\sum_{k_i\in K} (|k_i| - |p(k_i)|)}{\sum_{k_i\in K} (|k_i| - 1)}$$

Donde $p(k_i)$ es el conjunto de particiones que se crean de intersectar k_i con todas las entidades de R.

La Precision se calcula intercambiando el rol de las entidades *key* con las entidades *response* en la fórmula de arriba.

Intuitivamente, el Recall de MUC se puede interpretar como los enlaces que faltan en las entidades *response* (o uniones de entidades) para lograr formar las entidades *key*. La Precision de MUC se puede interpretar como la cantidad de enlaces incorrectos agregados que provocaron la unión de entidades o menciones incorrectos.

Según el trabajo de [45], esta es la métrica menos discriminativa; es decir, con la menor distinción entre links malos y muy malos. También, favorece la creación de entidades "sobre-dimensionadas"; es decir, entidades *response* que contienen las menciones de muchas entidades *key*.

B^3

B^3 [2] se propuso para aliviar los problemas de MUC. Es una métrica *entity-based* o fuertemente dependiente de los enlaces asignados a cada mención.

El Recall de B^3 se calcula como:

$$\text{Recall } = \frac{\sum_{k_i \in K} \sum_{r_j \in R} \frac{\left|k_i \cap r_j\right|^2}{|k_i|}}{\sum_{k_i \in K} |k_i|}$$

La Precision de B^3 se calcula intercambiando el rol de las entidades key con las entidades response dela ecuación de arriba.

El Recall de B^3 se puede interpretar como que tan bien se encuentra representada una entidad *key* en las entidades *response*; si la entidad *key* se encuentra repartida entre múltiples entidades *response*, el aporte de esa entidad al Recall será más bajo. La Precision de B^3 se puede interpretar como que tan precisa es cada entidad *response*; si la entidad *response* contiene menciones pertenecientes a varias entidades *key*, el aporte de esa entidad a la Precision será más bajo.

Según el estudio realizado en [45], esta métrica sufre de un efecto llamado *mention identification effect*, el cual hace que en ciertos casos esta métrica produzca resultados no confiables, y también en otros casos tiene resultados contraintuitivos y poco interpretables.

CEAF

CEAF [42] intenta hallar una asignación 1 a 1 entre entidades *key* y entidades *response*. Para esto se utiliza una métrica de similaridad entre entidades ϕ, y el algoritmo Kunh-Munkres para hallar la asignación óptimo (g^*) entre entidades *key* y *response*. Sea K^* las entidades *key* que estan incluidas en la asignación óptima, el Recall de CEAF se puede calcular como:

$$\text{Recall } = \frac{\sum_{k_i \in K^*} \phi\left(k_i, g^*\left(k_i\right)\right)}{\sum_{k_i \in K} \phi\left(k_i, k_i\right)}$$

La Precision de CEAF se puede calcular como:

$$\text{Precision } = \frac{\sum_{k_i \in K^*} \phi\left(k_i, g^*\left(k_i\right)\right)}{\sum_{R_i \in R} \phi\left(r_i, r_i\right)}$$

La métrica puede variar dependiendo de la función de similaridad ϕ utilizada. En la variante CEAF_m (*mention-based*), la similaridad es una función del número de menciones en común entre ambas entidades. En la variante CEAF_e (*entity-based*), la similaridad se calcula como $\phi\left(k_i, r_j\right) = \frac{2 \times \left|k_i \cap r_j\right|}{|k_i| + \left|r_j\right|}$.

Esta métrica también sufre del efecto de *mention identification* [45] y de resultados contraintuitivos en ciertos casos.

BLANC

BLANC [55] es otra métrica *link-based*, que consiste en una adaptación del índice Rand, utilizado comúnmente en la evaluación de la identificación de *clusters*. Sea C_k y C_r los conjuntos de enlaces de correferencias en las entidades *key* y *response* respectivamente. Sean N_k y N_r los conjuntos de enlaces de no-correferencia en las entidades *key* y *response*.

El Recall y Precision para los enlaces de correferencia se calculan como:

$$R_c = \frac{|C_k \cap C_r|}{|C_k|}, \quad P_c = \frac{|C_k \cap C_r|}{|C_r|}$$

El Recall y Precision para los enlaces de no-correferencia se calculan como:

$$R_n = \frac{|N_k \cap N_r|}{|N_k|}, \quad P_n = \frac{|N_k \cap N_r|}{|N_r|}$$

El Recall y Precision BLANC se calculan promediando el Recall y Precision de los enlaces de correferencia y no-correferencia

$$\text{Recall } = \frac{R_c + R_n}{2}, \quad \text{Precision } = \frac{P_c + P_n}{2}$$

BLANC también sufre el efecto de *mention identification* [45].

LEA

LEA (Link-Based Entity-Aware) [45] es una métrica propuesta con el objetivo de solventar el efecto de *mention identification*. Es una métrica *link-based*, pero aumentada con conocimiento de las entidades; considera que tan importante es una entidad y que tan bien resuelta está.

A grandes rasgos, LEA evalúa un conjunto de entidades de la siguiente forma:

$$\frac{\sum_{e_i \in E} (\text{ importance } (e_i) \times \text{ resolution-score } (e_i))}{\sum_{e_k \in E} \text{ importance } (e_k)}$$

Como medida de importancia de una entidad, se puede considerar su tamaño (importance $(e) = |e|$). Otras aplicaciones que utilicen el sistema de correferencias pueden definir otra métrica de importancia que se ajuste mejor a ese escenario, lo cual hace que esta métrica sea adaptable.

Se tiene que la entidad e con n menciones tiene $\text{link}(e) = n \times (n-1)/2$ enlaces de correferencia. El puntaje de resolución de la entidad k_i se computa como la fracción de enlaces resueltos de k_i:

$$\text{resolution-score}(k_i) = \sum_{r_j \in R} \frac{\text{link}(k_i \cap r_j)}{\text{link}(k_i)}$$

El Recall de LEA se computa como:

$$\text{Recall} = \frac{\sum_{k_i \in K} (|k_i| \times \sum_{r_j \in R} \frac{\text{link}(k_i \cap r_j)}{\text{link}(k_i)})}{\sum_{k_z \in K} |k_z|}$$

La Precision de LEA se computa como:

$$\text{Precision} = \frac{\sum_{r_i \in R} (|r_i| \times \sum_{k_j \in K} \frac{\text{link}(r_i \cap k_j)}{\text{link}(r_i)})}{\sum_{r_z \in R} |r_z|}$$

CoNLL Score

Como se mencionó a lo largo de esta sección, cada una de las métricas utilizadas tiene sus fortalezas y debilidades, por lo cual es difícil utilizar una única métrica como medida robusta de comparación de los sistemas. Por esto, en la conferencia CoNLL se planteó la utilización de un agregado de algunas de estas métricas.

El *CoNLL Score* se calcula como un promedio de los valores de F1 de las métricas MUC, B^3 y CEAF_e. En competencias de resolución de correferencia multilingüe, a su vez se toma el promedio de los *CoNLL Score* obtenidos para cada lenguaje.

D.2. Modelos

Los modelos propuestos para resolver la tarea se pueden clasificar de distintas formas. Algunos dividen la tarea en dos partes, detección de menciones y enlazado de menciones, y otros intentan resolver el problema en un solo paso.

Los primeros intentos de utilizar redes neuronales para resolver el problema de resolución de correferencias, dividen el problema en detección de menciones (*Mention Detection*) y enlazado de menciones (*Entity Linking*) [25,26,28,30,32,33,50,51,54,84]. Estos enfoques consideran todos los *spans* posibles dentro de un texto y hacen uso de una función de puntaje entrenada para decidir si el *span* es una mención o no. Luego, otra función de puntaje toma pares de menciones y decide si ambas menciones son correferentes.

El primer trabajo de esta línea es [32], que utiliza una combinación de capas *Long Short-Term Memory* (LSTM), mecanismos de atención y redes *feed-forward* para computar representaciones de los *spans* y entrenar las funciones de puntaje. Este modelo superó los resultados obtenidos por sistemas previos basados en reglas y vectores de características construidas manualmente en base a información sintáctica y semántica.

Trabajos subsecuentes incorporaron diversas mejoras al modelo propuesto por [32]. Con el éxito de los modelos de lenguaje, varios trabajos intentaron incorporar los mismos en la arquitectura para obtener mejores representaciones de los *spans* utilizados en las funciones de puntaje. El primero de estos fue [33], que utiliza el modelo ELMo [53] para obtener representaciones de los *tokens* de la secuencia previo a pasarlas por la capa LSTM del modelo original. Luego, [26] sustituye el modelo ELMo por BERT y obtiene mejores resultados. En [25] se propone el modelo SpanBERT, el cual posee una arquitectura similar a BERT pero fue entrenado específicamente para predecir el siguiente *span* en una secuencia de *tokens*, el cual mejoró aún más los resultados al incorporarse en la arquitectura previa.

En el trabajo de [54] se demuestra que entrenar sobre múltiples lenguajes, con el objetivo de obtener un sistema para la tarea de *Multilingual Coreference Resolution*, tiene el beneficio de que los resultados independientes para cada lenguaje mejoran. En este trabajo, se entrena un modelo similar al de [32] pero sobre el conjunto de datos CorefUD [48], que contiene anotaciones para 11 lenguajes. Entrenar sobre todos los lenguajes al mismo tiempo logra que se incremente en 1 punto el valor de la métrica F1 para el lenguaje español. Aunque esto no es una mejora significativa, para idiomas como el alemán, se obtienen mejoras de 9 puntos de F1. Otros modelos propuestos subsecuentemente [7,63,64] utilizan la estrategia de entrenar sobre todos los lenguajes al mismo tiempo para obtener sistemas multilingües.

Otras mejoras van en la línea de incorporar la noción de *Higher-order Inference* a la arquitectura [28, 33]. Este enfoque intenta incorporar información global sobre las entidades computadas o detectadas hasta el momento a la predicción de un enlace de correferencia entre dos menciones. Por ejemplo, el enfoque propuesto en [28] modifica la representación de una mención sumando información agregada de todas las menciones de su mismo *cluster* detectadas hasta el momento. Como se probó empíricamente en el trabajo de [84], el efecto obtenido por la inclusión de estos enfoques es insignificante, e incluso algunas veces perjudicial, por lo cual trabajos más recientes han descartado la inclusión de dicha información. Dicho efecto pasó desapercibido ya que dichos enfoques fueron agregados en conjunto con mejores modelos de lenguajes, los cuales sí traen mejoras sustanciales a los resultados.

Estos sistemas computan representaciones vectoriales de todos los posibles *spans*. Esto puede resultar perjudicial para el sistema en términos de consumo de memoria, ya que todas estas representaciones se deben mantener en memoria, y para la carga de

cómputo, ya que se deben computar las funciones de puntaje sobre todo el espacio de posibles *spans* de un texto. Para esto, se han desarrollado diversas técnicas de filtrado de *spans* [32, 33], como limitar el largo de los posibles *spans* a L *tokens*, retener solo los K *spans* con puntaje más alto de mención, evitar *spans* superpuestos y utilizar funciones de puntaje adicionales que permiten filtrar *spans* de mala calidad. Como se menciona en [30], estas técnicas siguen reteniendo una gran cantidad de posibles *spans*, lo cual resulta en altos costos de cómputo. Por eso, proponen remover las representaciones vectoriales de los *spans*, y utilizar representaciones especiales para los *tokens* de la secuencia, que representen si el *token* es el principio o el fin de una mención. Dichas representaciones de los *tokens* son usadas en conjunto dentro de las funciones de *scoring*. Este enfoque reduce significativamente el consumo de memoria, sin perjudicar significativamente los resultados obtenidos.

El trabajo [81] propone el modelo CorefQA, el cual tiene un enfoque alternativo al de [32] para el módulo de enlazado de menciones. En este, el problema se reformula dentro de un *framework* de *Question Answering* (QA); dado el documento sobre el que se está trabajando, se formula una pregunta como una mención que fue detectada previamente con el contexto que la rodea, y la respuesta esperada son los *spans* de texto correferentes a la mención de la pregunta dentro del documento. De forma más abstracta, se formula el problema como uno de extracción de *spans*. Este método tiene la peculiar ventaja de que se puede realizar *fine-tuning* sobre otros conjuntos de datos de QA, lo cual los autores prueban que es beneficioso para el modelo. A pesar de haber alcanzado los mejores resultados del estado del arte en su lanzamiento, otros trabajos argumentan que este modelo es computacionalmente costoso, ya que para cada potencial mención en el texto se debe hacer inferencia con un modelo grande de lenguaje [7, 51].

Con la explosión en popularidad y producción de modelos de lenguaje de gran escaka (LLM) para la generación de texto, ha habido interés por explotar dichos modelos para resolver esta tarea. Al momento de esta investigación, el único trabajo en esta línea es el de [7], que propone un cambio de arquitectura radical con respecto a los enfoques previos y explota un modelo de generación de texto para atacar el problema. Modelan el problema como uno de texto a texto o *seq2seq* y lo resuelven con un sistema *Transition Based*, en el cual el sistema recibe como entrada una porción del texto ya procesada, donde los *clusters* de correferencias se encuentran etiquetados y una porción nueva del texto sin ninguna etiqueta. Como salida, el modelo genera una tira de texto que contiene acciones; estas acciones pueden enlazar menciones encontradas en la nueva porción de texto con *clusters* identificados previamente o pueden crear nuevos *clusters* de correferencias. El sistema utiliza el modelo mT5 [85], en sus variantes de mayor tamaño, el cual es un LLM multilingüe. Los autores prueban que, entrenando el modelo sobre un conjunto de datos de correferencias en inglés, se pueden obtener muy buenos resultados para otros lenguajes en modalidad *zero-shot* o *few-shot*, principalmente debido al uso de un modelo multilingüe como base. Este enfoque innovador obtuvo los mejores resultados del estado del arte para el conjunto de datos CoNLL 2012.

Otro trabajo reciente de alta relevancia es el sistema CorPipe, presentado en las conferencias CRAC 2022 y CRAC 2023 [63, 64]. Este enfoque también emplea un módulo de *Mention Detection* y otro de *Entity Linking*, con la peculiaridad de que ambos módulos utilizan un LLM de base que es entrenado en paralelo para ambas tareas. En la primera versión del sistema, el LLM de base es utilizado para obtener una representación intermedia de los *tokens* de la porción de texto que se está procesando. Luego, dicha representación es pasada secuencialmente a dos cabezas específicas para cada módulo. El módulo de *Mention Detection* propone un sistema *Transition Based* para la detección de menciones en la secuencia, mientras que el módulo de *Entity*

Linking utiliza las representaciones calculadas y mécanismos de *self-attention* para predecir un puntaje de correferencia entre las menciones detectadas por el módulo anterior. Ambas cabezas, junto con el modelo base, son entrenadas al mismo tiempo. A su vez, en vez de utilizar todo el documento, el sistema emplea una ventana movil de contexto para procesar el documento entero, lo cual reduce altamente los requisitos computacionales del entrenamiento y la inferencia.

La segunda versión de este modelo fue presentada en CRAC 2023 [63]. Una de las principales mejoras involucra el uso de un LLM de mucho mayor tamaño. Una de las ventajas de utilizar un LLM más grande, además de obtener mejores representaciones de los *tokens*, es que en inferencia, se puede utilizar una ventana de contexto de mayor tamaño, ya que este modelo soporta un mayor contexto de entrada. Como demuestran los autores, aún entrenando con una ventana más pequeña, agrandar la ventana durante inferencia mejora los resultados significativamente. Este modelo nuevamente obtuvo los mejores resultados del estado del arte hasta el momento de escritura de este trabajo, superando ampliamente a los demás modelos presentados en la conferencia.

D.3. Conjuntos de datos

Existen conjuntos de datos para una pluralidad de lenguajes. Sin embargo, es de interés indagar sobre la existencia de conjuntos de datos para el lenguaje español.

El conjunto de datos con anotaciones de correferencias para Español más conocido es AnCora-CO [56]. El mismo cuenta con anotaciones de correferencias para el lenguaje Español y Catalán, además de anotaciones morfológicas, sintácticas y semánticas. Cada versión del conjunto cuenta con 400.000 palabras, extraídas de fuentes de noticias en el respectivo lenguaje. El conjunto de datos se encuentra disponible públicamente.

Uno de los problemas de AnCora-CO, y de los conjuntos de datos de otros lenguajes, es que el formato en el cual se detallan las anotaciones no se encuentra estandarizado. Esto dificulta el estudio de los datos y el desarrollo rápido de soluciones para la tarea de *Multilingual Coreference Resolution*. Por lo tanto, en los últimos años, el proyecto CorefUD [48] ha dedicado grandes esfuerzos a estandarizar las anotaciones de correferencias en múltiples lenguajes. El formato elegido es el de Universal Dependencies [15], el cual es un *framework* para la anotación de lenguajes y creación de *treebanks* en decenas de lenguajes. Como parte de los esfuerzos de CorefUD, se convirtió AnCora-CO desde su formato original al formato CorefUD. Esta nueva versión también se encuentra públicamente disponible.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

Expresiones regulares utilizadas en el Procesador de Salida

El módulo Procesador de Salida se encarga de procesar la salida cruda del LLM e instanciar la información extraída. Este procesamiento hace uso de múltiples expresiones regulares que permiten reconocer el texto generado por el LLM y obtener los datos necesarios para instanciar las clases correspondientes (nombre de la clase y sus argumentos). Por ejemplo, en el caso de una relación, se debe reconocer el tipo de relación y las entidades que une.

Las siguientes secciones explican las distintas expresiones regulares utilizadas por las dos versiones del Procesador de Salida implementadas.

En todos los casos, se hace uso de la funcionalidad de grupos de las expresiones regulares para extraer distintas secciones de un texto utilizando una única expresión regular. Las expresiones regulares son construidas utilizando el módulo `re` de Python.

E.1. Expresiones regulares utilizadas por el Procesador de Salida de Código

En el ejemplo E.1 se observa el bloque de código donde se definen las expresiones regulares utilizadas por el módulo Procesador de Salida de Código.

Ejemplo E.1: Expresiones regulares utilizadas por el módulo Procesador de Salida de Código.

```
ASSIGNMENTS_REGEX = re.compile(r"((entity|value|event|relation)\d+) =
    ↪ (\w+)\((.+)\)\n")
ARGUMENTS_PARSER = re.compile(r"(\w+)=('.+?'|\[.+?\]|.+?(?=,|\Z))")
```

La expresión `ASSIGNMENTS_REGEX` permite identificar en la salida del LLM las líneas que corresponden a la definición de una instancia de información. A su vez, los grupos de dicha expresión permiten extraer los siguientes datos:

- El nombre de la variable al que se está intentando asignar la instancia de información.
- El nombre de la clase que se está intentando instanciar.
- La lista de argumentos con los cuales se está intentando instanciar la clase.

El siguiente es un ejemplo de línea que cumple con esta expresión regular:

```
relation2 = Integrates(dep=entity5, gov=entity1)
```

La expresión `ARGUMENTS_PARSER` permite, a partir de la lista de argumentos extraída por la expresión `ASSIGNMENTS_REGEX`, obtener cada argumento en particular, separando su nombre y su valor, ya sea si el mismo es un valor de texto, una referencia a otra variable o una lista. En el ejemplo anterior, la expresión regular `ARGUMENTS_PARSER` permitiría obtener que hay un argumento de nombre `dep` y valor `entity5`, y otro de nombre `gov` y valor `entity1`.

E.2. Expresiones regulares utilizadas por el Procesador de Salida de Lenguaje Natural

En el ejemplo E.2 se observa el bloque de código donde se definen las expresiones regulares utilizadas por el módulo Procesador de Salida de Lenguaje Natural.

Ejemplo E.2: Expresiones regulares utilizadas por el módulo Procesador de Salida de Lenguaje Natural.

```
SPAN_REGEX = re.compile(r"^(.+) is an? (.+)\.?\Z")
RELATION_REGEX = re.compile(r"^Person (.+) (integrates|is family of|
    ↪ contacted|lives in) (Person|Organization|Location) (.+)\.?\Z")
EVENT_REGEX = re.compile(r"The trigger is (.+)\.?\Z")
SUBJECT_REGEX = r"^(<UNKNOWN>|Person .+(, Person .+)*)"
AUTH_REGEX = r"(<UNKNOWN>|Person .+|Organization .+)"
TIME_REGEX = r"(<UNKNOWN>|Time .+)"
CAUSE_REGEX = r"(<UNKNOWN>|Cause .+(, Cause .+)*)"
LOCATION_REGEX = r"(<UNKNOWN>|Location .+)"
RESPONSIBLE_REGEX = r"(<UNKNOWN>|Organization .+)"
DETENTION_ARGUMENTS_REGEX = re.compile(f"{SUBJECT_REGEX} (was|were)
    ↪ detained by {AUTH_REGEX} at {TIME_REGEX} and {LOCATION_REGEX}
    ↪ because of {CAUSE_REGEX}, and (was|were) transferred to {
    ↪ LOCATION_REGEX}\. The trigger is")
RELEASE_ARGUMENTS_REGEX = re.compile(f"{SUBJECT_REGEX} (was|were)
    ↪ released by {AUTH_REGEX} at {TIME_REGEX} and {LOCATION_REGEX},
    ↪ and (was|were) transferred to {LOCATION_REGEX}, under the
    ↪ responsibility of {RESPONSIBLE_REGEX}\. The trigger is")
TRANSFER_ARGUMENTS_REGEX = re.compile(f"{SUBJECT_REGEX} (was|were)
    ↪ transferred to {LOCATION_REGEX} at {TIME_REGEX} from {
    ↪ LOCATION_REGEX}\. The trigger is")
LIST_ITEM_REGEX = re.compile(r"(Person|Cause) (.+?)(?=(\Z|(, (Person|
    ↪ Cause))))")
ARG_REGEX = re.compile(r"(Cause|Person|Time|Location|Organization) (.+)
    ↪ \Z")
```

La instrucción de lenguaje natural solicita al LLM que defina statements que instancian la información a extraer. Se le instruye al LLM que se produzca un único statement por línea, y por lo tanto el Procesador de Salida de Lenguaje Natural procesa el texto generado por el LLM línea por línea.

Las expresiones `SPAN_REGEX`, `RELATION_REGEX` y `EVENT_REGEX` permiten determinar si la línea procesada corresponde a la definición de una entidad o valor (`SPAN_REGEX`), una relación (`RELATION_REGEX`) o un evento (`EVENT_REGEX`). A su vez, cada una de estas expresiones permite extraer información adicional utilizando grupos:

- `SPAN_REGEX` permite extraer el contenido del span y su tipo.
- `RELATION_REGEX` permite extraer las entidades participantes de la relación y el tipo de la relación.
- `EVENT_REGEX` permite extraer el texto del disparador del evento.

Las expresiones `SUBJECT_REGEX`, `AUTH_REGEX`, `TIME_REGEX`, `CAUSE_REGEX`, `LOCATION_REGEX` y `RESPONSIBLE_REGEX` permiten identificar el tipo de argumento y la entidad o valor involucrada para los distintos argumentos de eventos existentes. Estas expresiones se utilizan dentro de las expresiones `DETENTION_ARGUMENT_REGEX`, `RELEASE_ARGUMENTS_REGEX` y `TRANSFER_ARGUMENTS_REGEX` que permiten extraer completamente los eventos y sus argumentos. Si se identifica un statement como de tipo evento, luego se utilizan estas expresiones para determinar el tipo del evento y extraer sus argumentos y las entidades o valores que participan de esos argumentos.

Las expresiones `LIST_ITEM_REGEX` y `ARG_REGEX` se utilizan para continuar el procesamiento de los argumentos de eventos extraídos, permitiendo determinar si el contenido del argumento es una lista de entidades o valores o si es un valor singular, así como extraer el tipo de la entidad y su valor.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] Dimo Angelov. Top2vec: Distributed representations of topics. 2020.

[2] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer, 1998.

[3] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[4] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. Codekgc: Code language model for generative knowledge graph construction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(3):1–16, 2024.

[5] Samuel Blixen, Luciano Costabel, and Nilo Patiño. Una máquina (casi) perfecta : El sid como usina de inteligencia y soporte de la represión durante la dictadura militar. *Montevideo: FIC-Udelar*, 2018.

[6] Samuel Blixen, Nilo Patiño, Nadia Amesti, and Sofía Sánchez. Un modelo de guerra sucia: El rol operativo del ocoa en la represión. *Montevideo: FIC-Udelar*, 2018.

[7] Bernd Bohnet, Chris Alberti, and Michael Collins. Coreference resolution through a seq2seq transition-based system. *Transactions of the Association for Computational Linguistics*, 11:212–226, 2023.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[9] Pere-Lluís Huguet Cabot, Simone Tedeschi, Axel-Cyrille Ngonga Ngomo, and Roberto Navigli. Red-fm: a filtered and multilingual relation extraction dataset. *arXiv preprint arXiv:2306.09802*, 2023.

[10] Lautaro Cardozo Ramírez, Lía Emilia Rivero Cor, and Guillermo Zorrón Bordone. Luz : un sistema de búsqueda sobre los archivos de la última dictadura militar. *Montevideo: FING-Udelar*, 2021.

[11] Serge Chávez-Feria, Raúl García-Castro, and María Poveda-Villalón. Chowlk: from uml-based ontology conceptualizations to owl. In *European Semantic Web Conference*, pages 338–352. Springer, 2022.

[12] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[13] Linguistic Data Consortium. ACE 2005: Annotation tasks and specifications. `https://www.ldc.upenn.edu/collaborations/past-projects/ace/annotation-tasks-and-specifications`. Accessed: 2024-11-27.

[14] Linguistic Data Consortium. ACE 2005 multilingual training corpus. `https://catalog.ldc.upenn.edu/LDC2006T06`. Accessed: 2024-11-27.

[15] Marie-Catherine De Marneffe, Christopher D Manning, Joakim Nivre, and Daniel Zeman. Universal dependencies. *Computational linguistics*, 47(2):255–308, 2021.

[16] Antonella Dellanzo, Viviana Cotik, Daniel Yunior Lozano Barriga, Jonathan Jimmy Mollapaza Apaza, Daniel Palomino, Fernando Schiaffino, Alexander Yanque Aliaga, and José Ochoa-Luna. Digital surveillance in latin american diseases outbreaks: information extraction from a novel spanish corpus. *BMC bioinformatics*, 23(1):558, 2022.

[17] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[18] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] Hermenegildo Fabregat, Juan Martinez-Romo, and Lourdes Araujo. Overview of the diann task: Disability annotation task. In *IberEval@ SEPLN*, pages 1–14, 2018.

[20] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[21] Yucan Guo, Zixuan Li, Xiaolong Jin, Yantao Liu, Yutao Zeng, Wenxuan Liu, Xiang Li, Pan Yang, Long Bai, Jiafeng Guo, et al. Retrieval-augmented code generation for universal information extraction. *arXiv preprint arXiv:2311.02962*, 2023.

[22] Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90 % solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, 2006.

[23] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of the American medical informatics association*, 12(3):296–298, 2005.

[24] Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. Localized zeroth-order prompt optimization. *Advances in Neural Information Processing Systems*, 37:86309–86345, 2024.

[25] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

[26] Mandar Joshi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*, 2019.

[27] Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[28] Ben Kantor and Amir Globerson. Coreference resolution with entity equalization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, 2019.

[29] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 10, 2023.

[30] Yuval Kirstain, Ori Ram, and Omer Levy. Coreference resolution without span representations. *arXiv preprint arXiv:2101.00434*, 2021.

[31] Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart De Castilho, and Iryna Gurevych. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th international conference on computational linguistics: system demonstrations*, pages 5–9, 2018.

[32] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*, 2017.

[33] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018.

[34] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[35] Guozheng Li, Peng Wang, and Wenjun Ke. Revisiting large language models as zero-shot relation extractors. *arXiv preprint arXiv:2310.05028*, 2023.

[36] Peng Li, Tianxiang Sun, Qiong Tang, Hang Yan, Yuanbin Wu, Xuanjing Huang, and Xipeng Qiu. Codeie: Large code generation models are better few-shot information extractors. *arXiv preprint arXiv:2305.05711*, 2023.

[37] Zixuan Li, Yutao Zeng, Yuxin Zuo, Weicheng Ren, Wenxuan Liu, Miao Su, Yucan Guo, Yantao Liu, Xiang Li, Zhilei Hu, et al. Knowcoder: Coding structured knowledge into llms for universal information extraction. *arXiv preprint arXiv:2403.07969*, 2024.

[38] Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. A joint neural model for information extraction with global features. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7999–8009, 2020.

[39] Chengyuan Liu, Fubang Zhao, Yangyang Kang, Jingyuan Zhang, Xiang Zhou, Changlong Sun, Kun Kuang, and Fei Wu. Rexuie: A recursive method with explicit schema instructor for universal information extraction. *arXiv preprint arXiv:2304.14770*, 2023.

[40] Jie Lou, Yaojie Lu, Dai Dai, Wei Jia, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Universal information extraction as unified semantic matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13318–13326, 2023.

[41] Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. Unified structure generation for universal information extraction. *arXiv preprint arXiv:2203.12277*, 2022.

[42] Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 25–32, 2005.

[43] Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. Overview of tac kbp 2015 event nugget track. In *TAC*, 2015.

[44] Ying Mo, Jian Yang, Jiahao Liu, Shun Zhang, Jingang Wang, and Zhoujun Li. C-ICL: Contrastive in-context learning for information extraction. *arXiv preprint arXiv:2402.11254*, 2024.

[45] Nafise Sadat Moosavi and Michael Strube. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 632–642, 2016.

[46] Isabel Moreno, Ester Boldrini, Paloma Moreda, and M Teresa Romá-Ferri. Drugsemantics: a corpus for named entity recognition in spanish summaries of product characteristics. *Journal of biomedical informatics*, 72:8–22, 2017.

[47] Stephen Mutuvi, Antoine Doucet, Gaël Lejeune, and Moses Odeo. A dataset for multi-lingual epidemiological event extraction. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pages 4139–4144, 2020.

[48] Anna Nedoluzhko, Michal Novák, Martin Popel, Zdeněk Žabokrtskỳ, Amir Zeldes, and Daniel Zeman. Corefud 1.0: Coreference meets universal dependencies. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4859–4872, 2022.

[49] Mateo Nogueira. Construcción de herramientas para contribuir al análisis de los archivos de la ocoa. *Tesis de maestría. Universidad de la República (Uruguay). Facultad de Ingeniería.*, 2023.

[50] Shon Otmazgin, Arie Cattan, and Yoav Goldberg. F-coref: Fast, accurate and easy to use coreference resolution. *arXiv preprint arXiv:2209.04280*, 2022.

[51] Shon Otmazgin, Arie Cattan, and Yoav Goldberg. Lingmess: Linguistically informed multi expert scorers for coreference resolution. *arXiv preprint arXiv:2205.12644*, 2022.

[52] Hao Peng, Xiaozhi Wang, Feng Yao, Kaisheng Zeng, Lei Hou, Juanzi Li, Zhiyuan Liu, and Weixing Shen. The devil is in the details: On the pitfalls of event extraction evaluation. *arXiv preprint arXiv:2306.06918*, 2023.

[53] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[54] Ondřej Pražák, Miloslav Konopík, and Jakub Sido. Multilingual coreference resolution with harmonized annotations. *arXiv preprint arXiv:2107.12088*, 2021.

[55] Marta Recasens and Eduard Hovy. Blanc: Implementing the rand index for coreference evaluation. *Natural language engineering*, 17(4):485–510, 2011.

[56] Marta Recasens and M Antònia Martí. Ancora-co: Coreferentially annotated corpora for spanish and catalan. *Language resources and evaluation*, 44:315–345, 2010.

[57] Matthew Renze. The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356, 2024.

[58] Antonio Sabbatella, Andrea Ponti, Ilaria Giordani, Antonio Candelieri, and Francesco Archetti. Prompt optimization in large language models. *Mathematics*, 12(6):929, 2024.

[59] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[60] Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. Gollie: Annotation guidelines improve zero-shot information-extraction. *arXiv preprint arXiv:2310.03668*, 2023.

[61] Alessandro Seganti, Klaudia Firlag, Helena Skowronska, Michał Satława, and Piotr Andruszkiewicz. Multilingual entity and relation extraction dataset and model. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: Main volume*, pages 1946–1955, 2021.

[62] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[63] Milan Straka. \'ufal corpipe at crac 2023: Larger context improves multilingual coreference resolution. *arXiv preprint arXiv:2311.14391*, 2023.

[64] Milan Straka and Jana Straková. \'ufal corpipe at crac 2022: Effectivity of multilingual models for coreference resolution. *arXiv preprint arXiv:2209.07278*, 2022.

[65] Mariona Taulé, Maria Antònia Martí, and Marta Recasens. Ancora: Multilevel annotated corpora for catalan and spanish. In *Lrec*, volume 2008, pages 96–101, 2008.

[66] CodeGemma Team, Heri Zhao, Jeffrey Hui, Joshua Howland, Nam Nguyen, Siqi Zuo, Andrea Hu, Christopher A Choquette-Choo, Jingyue Shen, Joe Kelley, et al. Codegemma: Open code models based on gemma. *arXiv preprint arXiv:2406.11409*, 2024.

[67] Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. *arXiv preprint arXiv:2103.09330*, 2021.

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

[69] Amir Pouran Ben Veyseh, Javid Ebrahimi, Franck Dernoncourt, and Thien Huu Nguyen. Mee: A novel multilingual event extraction dataset. *arXiv preprint arXiv:2211.05955*, 2022.

[70] Marc Vilain, John D Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*, 1995.

[71] David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546*, 2019.

[72] Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. Deepstruct: Pretraining of language models for structure prediction. *arXiv preprint arXiv:2205.10475*, 2022.

[73] Jiahao Wang, Bolin Zhang, Qianlong Du, Jiajun Zhang, and Dianhui Chu. A survey on data selection for llm instruction tuning. *arXiv preprint arXiv:2402.05123*, 2024.

[74] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.

[75] Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. Instructuie: multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*, 2023.

[76] Xingyao Wang, Sha Li, and Heng Ji. Code4struct: Code generation for few-shot event structure prediction. *arXiv preprint arXiv:2210.12810*, 2022.

[77] Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. *arXiv preprint arXiv:2010.13415*, 2020.

[78] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[79] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[80] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*, 2023.

[81] Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. Corefqa: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, 2020.

[82] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*, 2023.

[83] Jun Xu, Mengshu Sun, Zhiqiang Zhang, and Jun Zhou. Chatuie: Exploring chat-based unified information extraction using large language models. *arXiv preprint arXiv:2403.05132*, 2024.

[84] Liyan Xu and Jinho D Choi. Revealing the myth of higher-order inference in coreference resolution. *arXiv preprint arXiv:2009.12013*, 2020.

[85] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[86] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[87] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint 2408.01800*, 2024.

[88] Hongbin Ye, Ningyu Zhang, Hui Chen, and Huajun Chen. Generative knowledge graph construction: A review. *arXiv preprint arXiv:2210.12714*, 2022.

[89] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

[90] Zdeněk Žabokrtskỳ, Miloslav Konopík, Anna Nedoluzhko, Michal Novák, Maciej Ogrodniczuk, Martin Popel, Ondřej Pražák, Jakub Sido, and Daniel Zeman. Findings of the second shared task on multilingual coreference resolution. In *Proceedings of the CRAC 2023 Shared Task on Multilingual Coreference Resolution*, pages 1–18, 2023.

[91] Zdeněk Žabokrtskỳ, Miloslav Konopík, Anna Nedoluzhko, Michal Novák, Maciej Ogrodniczuk, Martin Popel, Ondřej Pražák, Jakub Sido, Daniel Zeman, and Yilun Zhu. Findings of the shared task on multilingual coreference resolution. *arXiv preprint arXiv:2209.07841*, 2022.

[92] Kai Zhang, Bernal Jiménez Gutiérrez, and Yu Su. Aligning instruction tasks unlocks large language models as zero-shot relation extractors. *arXiv preprint arXiv:2305.11159*, 2023.

[93] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

[94] Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.

[95] Tong Zhu, Junfei Ren, Zijian Yu, Mengsong Wu, Guoliang Zhang, Xiaoye Qu, Wenliang Chen, Zhefeng Wang, Baoxing Huai, and Min Zhang. Mirror: A universal framework for various information extraction tasks. *arXiv preprint arXiv:2311.05419*, 2023.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de tablas

Índice de figuras

Índice de figuras

Esta es la última página.
Compilado el sábado 16 agosto, 2025.
http://iie.fing.edu.uy/