Universidad Mayor de San Andrés

Facultad de Ingeniería

Ingeniería Electrónica

Aplicación de técnicas de Machine Learning y Procesamiento de Lenguaje Natural para la extracción automática de datos bibliográficos en documentos académicos de la Universidad Mayor de San Andrés

Presentado por:	Rodrigo Orlando Iturralde Aliaga
Tipo de trabajo:	Proyecto de Grado
Tutor	Ing. Oscar Marcelo Poma Calle
Fecha:	2023

Agradecimientos

Gracias a los profesores que me han acompañado en este proceso de aprendizaje, a mis compañeros de clase, a mi familia y a mis amigos.

En especial, agradezco a mi tutor, el Ing. Marcelo Poma, por su orientación y apoyo en el desarrollo de este proyecto, asi como tambien al Ing. Freddy Valle por las directrices y recomendaciones para concretar este trabajo.

Resumen

La Universidad Mayor de San Andrés (UMSA) está digitalizando sus bibliotecas y creando una biblioteca virtual para almacenar los documentos académicos elaborados por sus estudiantes (proyecto de grado, tesis, etc). Sin embargo, aún hay miles de estos documentos pendientes de ser indexados y publicados en la biblioteca. El proceso de indexación es manual y requiere mucho tiempo. Este proyecto pretende demostrar que es posible utilizar Natural Language Processing (NLP) y Machine Learning (ML) para automatizar la extracción de datos bibliográficos de documentos y así acelerar el proceso de indexación. El objetivo es entrenar modelos de ML para identificar y extraer información como autor, título, año de publicación, palabras clave, etc. desde estos documentos.

En concreto se desarrollaron dos modelos: un modelo de Reconocimiento de Entidades Nombradas (NER) para extraer información desde la carátula de los documentos y un Modelo de Lenguaje Grande (LLM) para extraer palabras clave desde el resumen de los documentos.

Palabras clave: Reconocimiento de Entidades Nombradas, Modelo de Lenguaje Grande, Aprendizaje Automático, Procesamiento de Lenguaje Natural, Indexación de Documentos.

Abstract

The Universidad Mayor de San Andrés (UMSA) is digitizing its libraries and creating a virtual library to store the academic documents produced by its students (senior capstone projects, theses, etc). However, there are still thousands of these documents pending to be indexed and published in the library. The indexing process is manual and requires a lot of time. This project aims to demonstrate that it is possible to use Natural Language Processing (NLP) and Machine Learning (ML) to automate the extraction of bibliographic data from documents and thus accelerate the indexing process. The objective is to train ML models to identify and extract information such as author, title, year of publication, keywords, etc. from these documents.

Specifically, two models were developed: a Named Entity Recognition (NER) model to extract information from the cover of the documents and a Large Language Model (LLM) to extract keywords from the abstract of the documents.

Keywords: Named Entity Recognition, Large Language Model, Machine Learning, Natural Language Processing, Document Indexing.

LISTA DE FIGURAS

Figura 1 - Software de anotación PAWLS 3
Figura 2 - Metodología fundamental para la ciencia de datos. 7
Figura 3 - Estructura de DSpace. 9
Figura 4 - Pagina principal del repositorio DSpace implementado por la UMSA. 11
Figura 5 - Repositorio de la UMSA, vista individual de ítem 14
Figura 6 - Part-of-speech tagging implementado con una red neuronal. 17
Figura 7 - Linea de tiempo del NLP mostrando las diferentes estrategias usadas. 17
Figura 8 - Ejemplo de un sistema NER. 19
Figura 9 - Mirada de como una CNN funciona para tareas de NLP. 22
Figura 10 - NLP pipeline del modelo `es_core_news_lg` de spaCy. 25
Figura 11 - Ejemplo de uso del Entity Ruler 28
Figura 12 - Pipeline NLP especializado para Named-entity Recognition (NER) 28
Figura 13 - Arquitectura del modelo NER predefinido `spaCy.TransitionBasedParser.v2`. 30
Figura 14 - Ejemplo de un texto de entrada anotado con etiquetas IOB 32
Figura 15 - Ejemplo del texto de una carátula anotada en el formato esperado por spaCy. . 33
Figura 16 - Diseño general del sistema a desarrollar. 57
Figura 17 - Pipeline de SpaCy para la extracción de metadatos. 57
Figura 18 - Ejemplo de extracción de datos mediante *parsing* del archivo XML. 66
Figura 19 - Pandas Dataframe resultante del parsing realizado desde los archivos XML de metadatos. 67
Figura 20 - Extracción del texto de la carátula 67
Figura 21 - Ejemplo de la extracción del texto desde una carátula. 68
Figura 22 - Pandas Dataframe resultante de la extracción del texto de la carátula de todos los documentos recolectados. 69
Figura 23 - Pandas Dataframe resultante de la consolidación de los datos de los metadatos y los textos de las carátulas. 70
Figura 24 - Número de publicaciones por año 71
Figura 25 - Número de publicaciones por tipo (primera parte) 72
Figura 26 - Número de publicaciones por tipo (continuación) 73

Figura 27 - Ejemplo de documentos a descartar del conjunto inicial de documentos. Estos son descartados por no tener un formato de carátula convencional. 74
Figura 28 - Primera página de una muestra aleatoria de cada tipo de documento seleccionado 77
Figura 29 - Número de documentos por facultad, en el conjunto de datos final 78
Figura 30 - Número de documentos por tipo, en el conjunto de datos final 79
Figura 31 - Ejemplos del proceso de limpieza del texto 80
Figura 32 - Dataframe consolidado. El mismo contiene el texto de las carátulas y el texto de cada metadato. 81
Figura 33 - Modelo NER basado en reglas usado para generar los datos de entrenamiento 81
Figura 34 - Estado final de las métricas de entrenamiento. Métricas extraídas desde MLFlow. 84
Figura 35 - Métricas de entrenamiento 85
Figura 36 - Loss de todos los modelos mostrando la convergencia en todos ellos. 87
Figura 37 - Métricas de entrenamiento para cada modelo por entidad. 88
Figura 38 - Ejemplo de Prompt Zero-shot 93
Figura 39 - Ejemplo de Prompt One-shot 93
Figura 40 - Ejemplo de Prompt Few-shot 94
Figura 41 - Dataframe de metadatos inicial para el Ajuste Fino 95
Figura 42 - Dataframe resultante tras el proceso de limpieza 95
Figura 43 - Prompt usado durante el desarrollo del modelo LLM 97
Figura 44 - Configuración de LORA para el ajuste fino del modelo LLM 97
Figura 45 - Pseudocódigo para obtener las métricas ROUGE para cada modelo. 98
Figura 46 - Bosquejo de la aplicación de demostración. Vista de Formulario. 101
Figura 47 - Bosquejo de la aplicación de demostración. Vista de Salida de los Modelos . . 102
Figura 48 - Bosquejo de la aplicación de demostración. Vista de Metadatos en XML en formato Dublin Core 103
Figura 50 - Interfaz gráfica de la aplicación de demostración. Vista de formulario. 104
Figura 49 - Diagrama de secuencia de la aplicación de demostración 105
Figura 51 - Interfaz gráfica de la aplicación de demostración. Vista de Salida RAW de los Modelos 106

Figura 52 - Interfaz gráfica de la aplicación de demostración. Vista de Metadatos en Dublin Core 107
Figura 53 - Captura de pantalla de la aplicación tras haber extraido los metadatos y generado las palabras clave 110

LISTA DE TABLAS

Tabla 1 - Metadatos usados para describir a un item en el repositorio institucional de la UMSA 12
Tabla 2 - Entidades a reconocer por el modelo entrenado. 58
Tabla 3 - Selección de LLM basado en sus carácterísticas 60
Tabla 4 - Criterios de selección de datos y número de datos tras cada operación de filtrado. 76
Tabla 5 - Ejemplo de reglas definidas en el modelo NER auxiliar. Las reglas fueron basadas en los metadatos extraídos desde los XMLs 82
Tabla 6 - Porcentaje de división de los datos en conjunto de entrenamiento y conjunto de pruebas 83
Tabla 7 - División de los datos de entrenamiento 96
Tabla 8 - Comparación de los resultados obtenidos por el modelo ajustado con los obtenidos por otros modelos de generación de lenguaje 99

Glosario de abreviaturas y siglas

API Application Programming Interface (Interfaz de Programación de Aplicaciones)

BERT Bidirectional Encoder Representations from Transformers (Representaciones de Codificador Bidireccional de Transformadores)

BLEU Bilingual Evaluation Understudy (Estudio de Evaluación Bilingüe)

CNN Convolutional Neural Network (Red Neuronal Convolucional)

DCT DCMI Metadata Terms (Términos de Metadatos de DCMI)

GPT-3 Generative Pre-trained Transformer 3 (Transformador Generativo Pre-entrenado 3)

JSON JavaScript Object Notation (Notación de Objetos de JavaScript)

LLM Large Language Model (Modelo de Lenguaje Grande)

LORA Low-Resource Adaptation (Adaptación de Recursos Bajos)

LSTM Long Short-Term Memory (Memoria Larga a Corto Plazo)

ML Machine Learning (Aprendizaje Automático)

NER Named Entity Recognition (Reconocimiento de Entidades Nombradas)

NLP Natural Language Processing (Procesamiento de Lenguaje Natural)

OCR Optical Character Recognition (Reconocimiento Óptico de Caracteres)

PEFT Parameter-Efficient Fine-Tuning (Ajuste Fino Eficiente de Parámetros)

POS Part-of-Speech (Etiquetado Gramatical)

ROUGE Recall-Oriented Understudy for Gisting Evaluation (Estudio de Evaluación Orientado al Recuerdo para Resumen)

SVM Support Vector Machine (Máquina de Vectores de Soporte)

UMSA Universidad Mayor de San Andrés

XML Extensible Markup Language (Lenguaje de Marcado Extensible)

Glosario de términos

Ajuste fino: O también conocido como *fine-tuning* en inglés, es el proceso de ajustar los parámetros de un modelo de aprendizaje automático para que se ajuste a un conjunto de datos específico.

Aprendizaje automático: O también conocido como *Machine Learning* en inglés, es un subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender a partir de datos.

Arquitectura de red neuronal: Estructura de una red neuronal que define cómo se conectan las neuronas y cómo se propagan las señales a través de la red.

Backend: Parte de un sistema de software que no interactúa directamente con los usuarios finales.

BERT: Bidirectional Encoder Representations from Transformers, es un modelo de lenguaje grande que utiliza una arquitectura de codificador de transformadores bidireccionales.

Biblioteca virtual: Repositorio digital que almacena y proporciona acceso a documentos digitales, como libros, tesis, artículos, etc.

BLEU: Es una métrica de evaluación de traducciones automáticas.

Convolutional Neural Network: Ver *Redes neuronales convolucionales*.

DataFrame: Es una estructura de datos bidimensional que se utiliza para almacenar y manipular datos en Python.

Datos de entrenamiento: Conjunto de datos que se utiliza para entrenar un modelo de aprendizaje automático. Los datos de entrenamiento consisten en ejemplos de entrada y salida que se utilizan para ajustar los parámetros del modelo.

Documento académico: Documento que se produce en el contexto de la educación superior, como tesis, proyectos de grado, memorias laborales, etc.

DSpace: Software de código abierto que permite a las instituciones gestionar repositorios digitales.

Entrenamiento (en el contexto de la ciencia de datos): Proceso de ajustar los parámetros de un modelo de aprendizaje automático para que se ajuste a los datos de entrenamiento.

Epoch o Época: Unidad de tiempo en el entrenamiento de un modelo de aprendizaje automático.

Una época es una pasada completa a través de todos los datos de entrenamiento.

Evaluación (en el contexto de la ciencia de datos): Proceso de medir el rendimiento de un modelo de aprendizaje automático en un conjunto de datos de prueba.

Extracción de datos bibliográficos: Proceso de identificar y extraer los datos bibliográficos de un documento, como el autor, el título, el año de publicación, etc.

F1-score: Es una métrica de evaluación de modelos de clasificación que combina la precisión y el recall en una sola métrica.

Few-shot prompt: Frase o pregunta que se utiliza para iniciar una conversación con un modelo de lenguaje grande proporcionando pocos ejemplos como referencia.

Fine-tuning: Ver *Ajuste fino*.

Frontend: Parte de un sistema de software que interactúa con los usuarios finales.

GPT-3: Generative Pre-trained Transformer 3, es un modelo de lenguaje grande que utiliza una arquitectura de decodificador de transformadores.

Hiperparámetros: Parámetros de un modelo de ML que se ajustan antes del entrenamiento para controlar el comportamiento del modelo durante el entrenamiento. Por ejemplo, la tasa de aprendizaje, el tamaño del lote, el número de épocas, etc.

Indexación: Proceso de registrar un documento en un repositorio digital junto con sus datos bibliográficos.

JSON: JavaScript Object Notation, es un formato de intercambio de datos ligero que se utiliza para transmitir datos entre un servidor y un cliente.

Large Language Model: Ver *Modelo de lenguaje grande*.

Layout: Ver Maquetación.

LORA: Low-Resource Adaptation, es una técnica de ajuste fino de modelos de lenguaje grande que utiliza una cantidad reducida de parámetros entrenables.

Loss: Pérdida, es una métrica de evaluación de modelos de aprendizaje automático que mide la diferencia entre las predicciones del modelo y los valores reales.

Maquetación: Disposición del texto e imágenes en las páginas de un documento.

Metadatos: Datos que describen otros datos. En el contexto de un repositorio digital, los metadatos describen el contenido y origen de los documentos.

Mistral: Es un modelo de lenguaje grande que utiliza una arquitectura de codificador de transformadores bidireccionales.

MMLU (benchmarks): Es un conjunto de datos de evaluación de modelos de lenguaje grande que se utiliza para comparar el rendimiento de diferentes modelos de lenguaje grande.

Modelado (en el contexto de la ciencia de datos): Proceso de entrenar y evaluar modelos de aprendizaje automático para resolver un problema específico.

Modelo de lenguaje grande: O también conocido como *Large Language Model* en inglés, es un modelo de aprendizaje automático que se entrena en una gran cantidad de datos de texto para predecir la siguiente palabra en una secuencia de texto.

MT-BENCH: Es un conjunto de datos de evaluación de traducción automática que se utiliza para comparar el rendimiento de diferentes sistemas de traducción automática.

Named Entity Recognition: Ver *Reconocimiento de entidades nombradas*.

Natural Language Processing: Ver *Procesamiento de Lenguaje Natural*.

One-shot prompt: Frase o pregunta que se utiliza para iniciar una conversación con un modelo de lenguaje grande proporcionando un solo ejemplo como referencia.

Optical Character Recognition: Ver *Reconocimiento óptico de caracteres*.

Palabras clave: Términos o frases que resumen el contenido de un documento y que se utilizan para facilitar la búsqueda y recuperación de información.

Parámetros entrenables: Parámetros de un modelo de lenguaje grande que se ajustan durante el entrenamiento para mejorar su rendimiento en una tarea específica.

PAWLS: Plataforma de anotación de texto para el aprendizaje automático.

PDF: Portable Document Format, es un formato de archivo que se utiliza para presentar e intercambiar documentos de forma fiable, independientemente del software, hardware o sistema operativo.

PEFT: Parameter-Efficient Fine-Tuning, es una técnica de ajuste fino de modelos de lenguaje grande que utiliza una cantidad reducida de parámetros entrenables.

PETAENG: Modalidad de graduación de la Universidad Mayor de San Andrés.

Pipeline de procesamiento de lenguaje natural: Conjunto de componentes que se utilizan para procesar un texto, como tokenizadores, etiquetadores gramaticales, analizadores de sentimientos, etc.

POS: Part-of-Speech, etiquetado gramatical. Es una técnica de procesamiento de lenguaje natural que asigna a cada palabra en un texto una categoría gramatical, como sustantivo, verbo, adjetivo, etc.

Precision: Es una métrica de evaluación de modelos de clasificación que mide la proporción de verdaderos positivos entre todos los elementos que el modelo clasificó como positivos.

Procesamiento de Lenguaje Natural: O también conocido como *Natural Language Processing* en inglés, es un subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras entender y generar lenguaje humano.

Prompt: Frase o pregunta que se utiliza para iniciar una conversación con un modelo de lenguaje grande.

PyTorch: Es una librería de aprendizaje automático de código abierto que se utiliza para desarrollar y desplegar modelos de lenguaje grande.

Recall: Es una métrica de evaluación de modelos de clasificación que mide la proporción de verdaderos positivos que el modelo clasificó correctamente entre todos los elementos que son realmente positivos.

Reconocimiento óptico de caracteres: O también conocido como *Optical Character Recognition* en inglés, es una técnica de procesamiento de imágenes que se utiliza para convertir imágenes de texto en texto digital.

Reconomiento de entidades nombradas: O también conocido como *Named Entity Recognition* en inglés, es una técnica de procesamiento de lenguaje natural que se utiliza para identificar y clasificar entidades en un texto, como nombres de personas, organizaciones, lugares, fechas, etc.

Redes neuronales convolucionales: O también conocidas como *Convolutional Neural Networks*

en inglés, son un tipo de red neuronal que se utiliza para procesar datos de imágenes y vídeo. También, ultimamente se utilizan para procesar datos de texto debido a su capacidad de aprender dependencias a largo plazo, es decir, su capacidad de aprender dependencias entre palabras que están separadas por muchas otras palabras.

Repositorio digital: Sistema que almacena y proporciona acceso a documentos digitales.

Repositorio institucional: Repositorio digital que almacena y proporciona acceso a documentos académicos producidos por una institución, como tesis, artículos, informes, etc.

RNN: Recurrent Neural Network, es un tipo de red neuronal que se utiliza para procesar datos secuenciales, como texto, audio y vídeo.

ROUGE: Recall-Oriented Understudy for Gisting Evaluation, es una métrica de evaluación de resúmenes automáticos.

Sistema de extracción de datos bibliográficos basado en maquetación: Sistema que utiliza técnicas de procesamiento de imágenes y reconocimiento de caracteres para identificar los elementos de los datos bibliográficos en un documento.

Sistema de extracción de datos bibliográficos basado en texto: Sistema que utiliza técnicas de procesamiento del lenguaje natural para identificar los elementos de los datos bibliográficos en un documento.

spaCy: Es una librería de procesamiento de lenguaje natural de código abierto para Python.

Token: Unidad de texto que se utiliza como entrada para un modelo de lenguaje grande, como una palabra, un subconjunto de una palabra o un carácter.

Tokenización: Proceso de dividir un texto en tokens.

Tokenizer: Componente de un pipeline de procesamiento de lenguaje natural que se utiliza para dividir un texto en tokens.

Trabajo de grado: Documento académico que un estudiante debe presentar para obtener un título académico, como una tesis, un proyecto de grado, una memoria laboral, etc.

Transformer: Es una arquitectura de red neuronal que se utiliza para procesar datos secuenciales y que es capaz de aprender dependencias a largo plazo.

XML: Extensible Markup Language, es un lenguaje de marcado que se utiliza para almacenar y transmitir datos.

Zero-shot prompt: Frase o pregunta que se utiliza para iniciar una conversación con un modelo de lenguaje grande sin proporcionar ejemplos de referencia.

INDICE DE CONTENIDOS

1 Marco referencial **1**

1.1 Introducción 1

1.1.1 Trabajos relacionados 2

1.2 Objetivos 4

1.2.1 Objetivo general 4

1.2.2 Objetivos específicos 4

1.3 Límites y alcances 5

1.3.1 Límites: 5

1.3.2 Alcances: 5

1.4 Metodología 6

1.5 Justificación 6

1.5.1 Justificación social 6

1.5.2 Justificación económica 6

1.5.3 Justificación académica 6

2 Marco teórico **7**

2.1 Metodología Fundamental para la Ciencia de Datos 7

2.2 Repositorio institucional DSpace 8

2.2.1 Metadatos 10

2.2.2 Mecanismos de busqueda y recuperación de información en DSpace 10

2.2.3 Uso del repositorio DSpace por la UMSA 11

2.3 Procesamiento de Lenguaje Natural 13

2.3.1 Named-entity Recognition 18

2.3.2 Extracción de palabras clave 22

2.4 Spacy 23

2.4.1 Pipelines en spaCy 24

2.4.2 Pipeline de NER basado en Entity Ruler 25

2.4.3 Pipeline de NER basado en ML 27

2.4.4 Estructuras de datos en spaCy 34

2.5 Modelos de Lenguaje Grande 34

2.5.1 Clasificación de LLMs 36
2.5.2 Ingeniería de prompts 37
2.5.3 Ajuste fino 38
2.5.4 Evaluación de LLMs 41
2.5.5 Benchmarks 42
2.6 Mistral-7B-Instruct-v0.1 44
2.7 Herramientas de software para el desarrollo de proyectos de Machine Learning . . . 45
2.7.1 Python 45
2.7.2 Jupyter Notebooks 46
2.7.3 DataSpell 47
2.7.4 OpenCV 48
2.7.5 Tesseract 49
2.7.6 Pandas 50
2.7.7 Huggingface 51
2.7.8 MLFlow 52

3 Marco práctico **53**
3.1 Etapa 1: Comprensión del negocio 53
3.1.1 Requerimientos 54
3.2 Etapa 2: Enfoque analítico 56
3.2.1 Descripcion de los subsistemas a desarrollar 57
3.3 Subsistema de extracción de metadatos 62
3.3.1 Etapa 3: Requisitos de datos 62
3.3.2 Etapa 4: Recopilación de datos 64
3.3.3 Etapa 5: Comprensión de datos 69
3.3.4 Etapa 6: Preparación de datos 75
3.3.5 Etapa 7: Modelado (1era iteración) 83
3.3.6 Etapa 8: Evaluación (1era iteración) 85
3.3.7 Etapa 7: Modelado (2da iteración) 86
3.3.8 Etapa 8: Evaluación (2da iteración) 89
3.4 Subsistema de generación de palabras clave 89
3.4.1 Etapa 3: Requisitos de datos 89

3.4.2 Etapa 4: Recopilación de datos 90
3.4.3 Etapa 5: Comprensión de datos 91
3.4.4 Etapa 6: Preparación de datos 92
3.4.5 Etapa 7: Modelado 96
3.4.6 Etapa 8: Evaluación 98
3.5 Elaboración de la aplicación de demostración 99
3.5.1 Diseño de la aplicación de demostración 99
3.5.2 Implementación de la aplicación 104

4 Análisis de resultados 108
4.1 Subsistema de extracción de metadatos 108
4.2 Subsistema de generación de palabras clave 109
4.3 Pruebas de la aplicación de demostración 109

5 Conclusiones, prospectiva y recomendaciones 111
5.1 Conclusiones 111
5.2 Prospectiva 111
5.3 Recomendaciones 112

Referencias 114

Apéndice A - Código fuente del proyecto 117

Apéndice B - Manual de uso de los modelos de ML desarrollados 118
Modelo de extracción de metadatos de la carátula 118
Modelo de generación de palabras clave 119

Apéndice C - Manual de usuario de la aplicación web de demostración 121
Requerimientos de la aplicación 121
Instalación de la aplicación 121
Ejecución de la aplicación 122
Uso de la aplicación 122

1 Marco referencial

1.1 Introducción

La Universidad Mayor de San Andrés (UMSA) se encuentra en proceso de digitalizar sus bibliotecas, lo que incluye la publicación de los trabajos de grado realizados por sus estudiantes en una biblioteca virtual[1].

La creación de este repositorio digital ha supuesto un gran avance para la institución, ya que ha permitido que miles de documentos sean fácilmente accesibles por estudiantes y docentes. Sin embargo, aún quedan miles de trabajos de grado pendientes de ser indexados[2] y publicados en la nueva biblioteca.

La demora en la publicación de estos documentos se debe a que el proceso de indexación de los trabajos de grado es manual y requiere de mucho tiempo - se estima que tomará cerca de 2 años más. Uno de los impedimentos para automatizar este proceso es la carencia de un software que pueda obtener los datos bibliográficos con los cuales indexar los documentos. Dichos datos son obtenibles desde el mismo documento, por lo que el desarrollo de un extractor de datos bibliográficos sería una solución muy efectiva.

El presente proyecto tiene como objetivo demostrar que el Procesamiento de Lenguage Natural (NLP) y el Machine Learning (ML) puede usarse para este propósito. Para ello, se propone entrenar modelos de ML a partir del conjunto de trabajos de grado que ya han sido indexados en el repositorio universitario. El modelo se entrenará para identificar y extraer los datos bibliográficos de los documentos, como ser: el autor, el título, el año de publicación, etc.

Este proyecto tiene un gran potencial para contribuir a la automatización del proceso de indexación de documentos en el repositorio de la UMSA. Cumpliendo con las metas planteadas, se podría automatizar el proceso de indexación de los miles de trabajos de grado pendientes de registro, lo que permitiría que estos sean accesibles a la comunidad universitaria más prontamente.

[1] La biblioteca virtual de la UMSA esta basado en el repositorio de código abierto DSpace (ver más en la Sección 2.2). La misma se encuentra disponible en la dirección https://repositorio.umsa.bo/

[2] La indexación de los documento consiste en el registro del mismo en el repositorio universitario junto a sus datos bibliográficos. Para ello durante el registro se llena un formulario en el cual se consignan todos los datos bibliográficos (autor, título, etc). Estos datos sirven de índice para que los documentos puedan ser fácilmente hallados por los usuarios del sistema mediante el sistema de búsqueda que la plataforma provee.

Asimismo, el resultado del proyecto podría ser un referente para que otros proyectos de digitalización puedan ser automatizados, no solo en bibliotecas de universidades, sino también en otros tipos de instituciones, como hospitales, juzgados, empresas, etc.

1.1.1 Trabajos relacionados

La mayoría de los sistemas de extracción de datos bibliográficos se basan en la maquetación, o *layout*[3], de los documentos. Estos sistemas utilizan técnicas de procesamiento de imágenes y reconocimiento de caracteres para identificar los elementos de los datos bibliográficos, como títulos, autores, afiliaciones y referencias basados en la ubicación de estos textos en las páginas de los documentos.

Algunos ejemplos de sistemas de extracción de datos bibliográficos basados en *layouts* son:

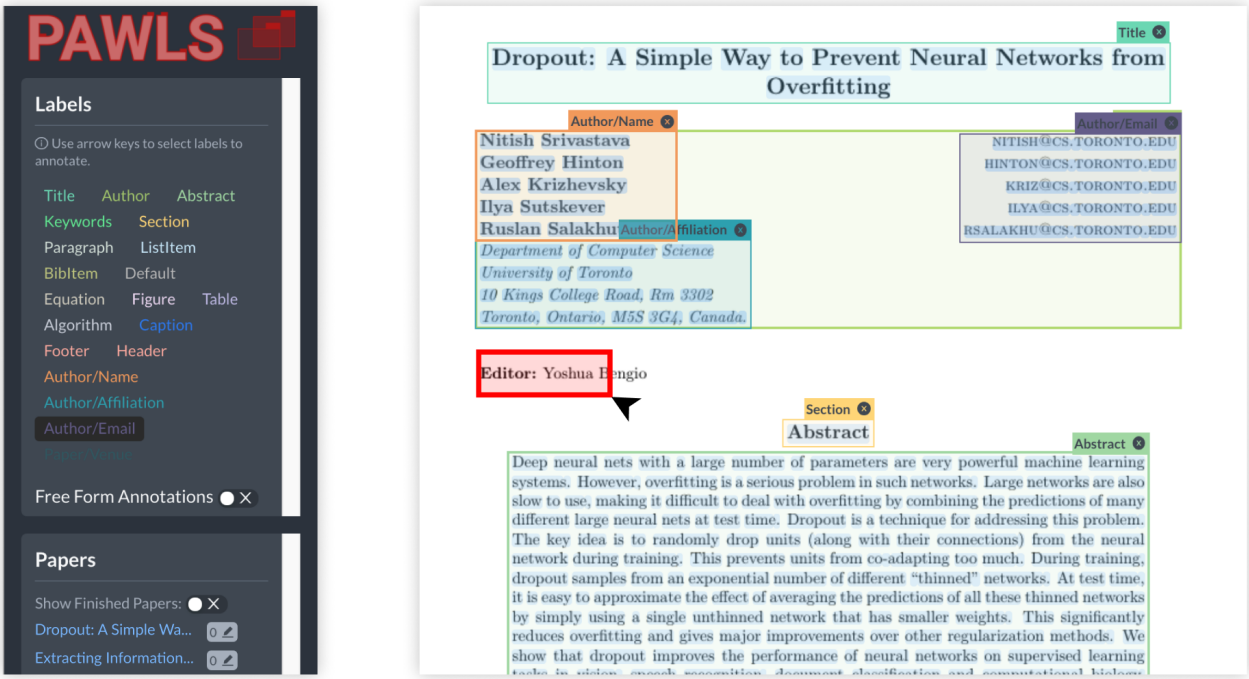
- **GROBID:** Es un software de código abierto que se utiliza para extraer datos bibliográficos de documentos científicos. GROBID utiliza un enfoque basado en maquetación para identificar los elementos de los datos bibliográficos. (Grobid, s.f.).
- **Mendeley Desktop**: Es una aplicación de escritorio que se utiliza para gestionar referencias bibliográficas. Una de sus capacidades es la de inferir los datos bibliográficos al momento de importarlos al programa. Para esto utiliza un algoritmo basado en Support Vector Machines (SVM) para extraer datos bibliográficos del texto del encabezado de los documentos. (Granitzer et al., 2012).
- **LiLT:** Es un sistema de extracción de datos bibliográficos basado en un modelo de aprendizaje profundo. LiLT utiliza un enfoque basado en maquetación para identificar los elementos de los datos bibliográficos (Wang et al., 2022).
- **LayoutLMv3:** Es un sistema de extracción de datos bibliográficos basado en un modelo de aprendizaje profundo. LayoutLMv3 utiliza un enfoque basado en maquetación para identificar los elementos de los datos bibliográficos. (Huang et al., 2022)

Estos sistemas son capaces de extraer los datos bibliográficos con un alto grado de precisión, sin embargo, su entrenamiento requiere la construcción de una gran cantidad de datos anotados, lo cual

[3]Maquetación o layout de los documentos se refiere a la disposición del texto e imágenes en las páginas del documento.

es un proceso que puede tomar muchísimo tiempo hacer. En la Figura 1, se muestra un ejemplo de cómo se anota cada dato de entrenamiento para este tipo de sistemas. Por el gran tiempo y esfuerzo que anotar los documentos representa, usar estos modelos no siempre es viable.

Figura 1 - Software de anotación PAWLS



1 Control Panel

Annotators can select label types, check labeling status, and add comments for the labeling task.

2 Labeling Canvas

Directly create annotations over PDF documents. To maximize efficiency, annotators create rectangular bounding boxes, with the contained tokens automatically selected and assigned to the label. Freeform bounding boxes without textual content are also supported.

Fuente: Neumann et al. (2021).

Una alternativa a los sistemas de extracción de datos bibliográficos basados en *layouts* son los sistemas basados en texto. Estos sistemas utilizan técnicas de procesamiento del lenguaje natural para identificar los elementos de los datos bibliográficos en el texto del documento.

Algunos ejemplos de sistemas de extracción de datos bibliográficos basados en texto son:

- **Sistemas basados en Named Entity Recognition (NER):** Un proyecto relevante a destacar es el de la National Library Board (NLB) de Singapur, que implementó un sistema de Reconocimiento de Entidades Nombradas (NER) con técnicas de Procesamiento del Lenguaje Natural (NLP) para extraer nombres de personas, organizaciones y lugares de artículos en sus plataformas

Infopedia y HistorySG. La extracción automatizada de entidades permitió a los usuarios buscar recursos relacionados dentro de las vastas colecciones de la biblioteca (Goh, 2017).

Este proyecto comparte similitudes con el presente en su objetivo de automatizar la extracción de información de documentos usando técnicas de procesamiento del lenguaje natural. Si bien se centra en la extracción de entidades generales, demuestra la viabilidad de aplicar técnicas similares para la extracción de datos bibliográficos específicos.

- **Sistemas basados en Modelos de Lenguaje Grande (LLMs):** KeyBERT y KeyLLM son ejemplo de ellos. Son librerías para Python las cuales permiten la extracción de palabras clave desde un texto dado mediante el uso de Modelos de Lenguaje Grande(Grootendorst, s.f.).

Estos sistemas son capaces de extraer datos bibliográficos de documentos con un grado de precisión comparable a los sistemas basados en *layouts*. Sin embargo, requieren datos de entrenamiento puramente basados en texto, lo cual es muchas veces más fácil y rápido de construir.

1.1.1.1 Solución propuesta

Se descarta el uso de los modelos basados en *layouts* debido a que la construcción de los datos de entrenamiento necesarios requeriría meses. Por el contrario, la construcción de datos de entrenamiento para modelos basados en texto es viable y se estima que puede realizarse en unos días. Por lo tanto, se propone desarrollar un sistema basado en modelos de ML cuyos datos de entrenamiento estén puramente basados en texto, siendo estos construídos a partir de los documentos ya indexados en el repositorio universitario.

1.2 Objetivos

1.2.1 Objetivo general

- Aplicar técnicas de Machine Learning y Procesamiento de Lenguaje Natural para extraer datos bibliográficos de documentos académicos de la Universidad Mayor de San Andrés.

1.2.2 Objetivos específicos

- Investigar y seleccionar técnicas de NLP y ML apropiados para la extracción de datos bibliográficos desde documentos de la UMSA, basándose en la literatura y prácticas en la

industria.

- Construir un conjunto de datos para el entrenamiento y evaluación de los modelos, basado en los datos disponibles en el repositorio universitario.
- Desarrollar los modelos de ML.
- Evaluar el rendimiento de los modelos en términos de métricas como ser Precisión, Recall y F1-score.
- Documentar y analizar los resultados obtenidos.

1.3 Límites y alcances

1.3.1 Límites:

- **Enfoque en los modelos de ML:** El proyecto se concentra en la creación y entrenamiento de los modelos de ML para la extracción de datos bibliográficos. No se desarrollará una interfaz gráfica completa ni se integrará con los sistemas existentes de la UMSA. La interfaz proporcionada será básica y servirá únicamente como herramienta de demostración.
- **Funcionalidad limitada:** El proyecto solo automatiza la extracción de datos bibliográficos. No abarca otros aspectos del proceso de registro de documentos académicos, como el envío de documentos y metadatos al repositorio universitario.

1.3.2 Alcances:

- **Automatización de la extracción de datos:** El proyecto automatiza la extracción de información como autores, título, fecha de publicación, palabras clave, etc., de los documentos académicos.
- **Parte de un sistema integral:** Los modelos de ML desarrollados formarán parte de un sistema futuro que automatizará el proceso completo de indexación de documentos.

1.4 Metodología

1.5 Justificación

1.5.1 Justificación social

Este proyecto tiene el potencial de tener un impacto significativo en la eficiencia de los procesos académicos y de investigación. Al automatizar la extracción de metadatos bibliográficos, la Universidad puede agilizar la indexación y publicación de documentos de fin de grado, optimizando el acceso y la búsqueda de información, crucial para estudiantes e investigadores.

1.5.2 Justificación económica

Esta proyecto es parte de una iniciativa que promete generar ahorros sustanciales en términos de recursos humanos y tiempo. Al automatizar el proceso de indexación de documentos académicos, se eliminará la necesidad de que personal dedique cientos de horas a esta tarea manual, lo que implica una disminución significativa de costos operativos asociados.

Esto se traducirá en un impacto económico positivo para la Universidad Mayor de San Andrés, al optimizar recursos, reducir costos y mejorar la productividad de todo el proceso de gestión documental y académica.

1.5.3 Justificación académica

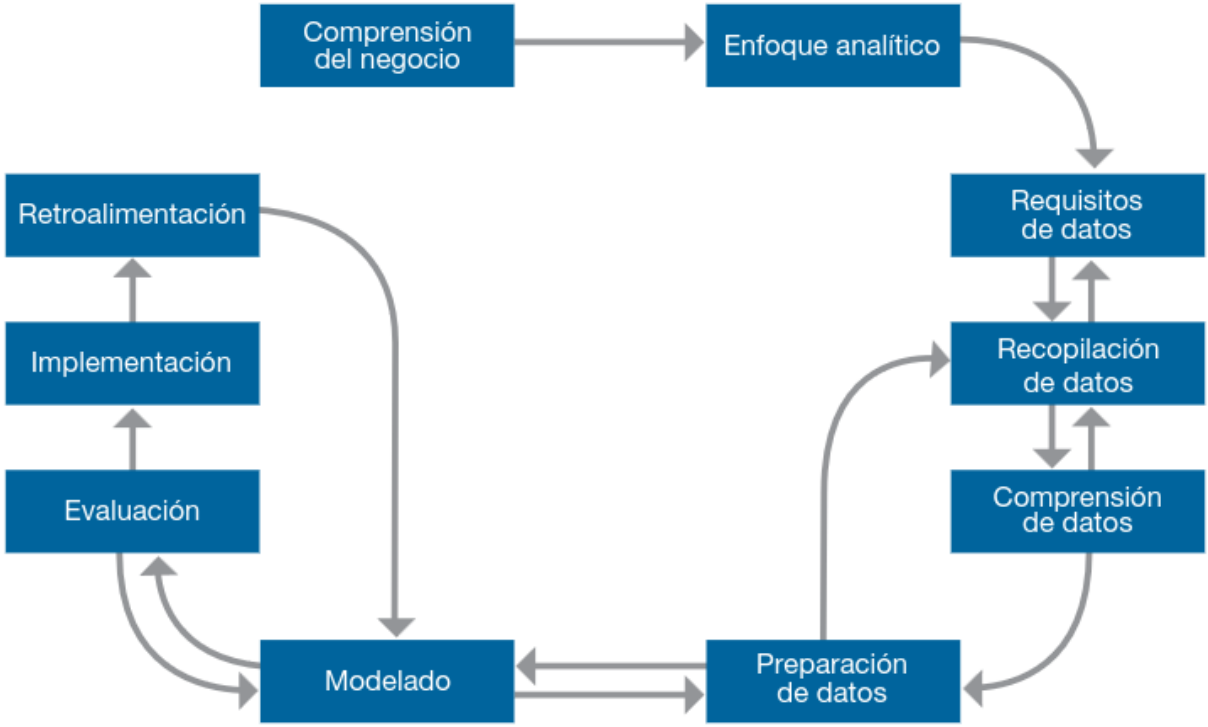
En el entorno académico y empresarial de nuestra región, la aplicación práctica del Procesamiento del Lenguaje Natural y el Machine Learning se encuentran en una etapa incipiente o, en muchos casos, son prácticamente inexistentes. A diferencia de otros países, donde estas tecnologías se han convertido en herramientas indispensables para mejorar la eficiencia de procesos, en nuestro país, todavía no se han aprovechado plenamente. Este contexto subraya la necesidad de demostrar la aplicabilidad de estos conceptos en proyectos reales. El presente proyecto no solo contribuirá a llenar este vacío tecnológico, sino que también abrirá nuevas perspectivas para la innovación y la eficiencia en el ámbito académico y empresarial de la región.

2 Marco teórico

2.1 Metodología Fundamental para la Ciencia de Datos

La Metodología Fundamental para la Ciencia de Datos es un marco de trabajo para la ciencia de datos que se basa en las mejores prácticas de la industria y en la experiencia de IBM en el campo (IBM, 2015). La metodología se divide en 10 etapas, las cuales se muestran en la Figura 2.

Figura 2 - Metodología fundamental para la ciencia de datos.



Fuente: Metodología Fundamental para la Ciencia de Datos (IBM, 2015).

Cada etapa de la metodología se describe a continuación:

- **Etapa 1 - Comprensión del negocio:** En esta etapa se identifican los objetivos del negocio y los requerimientos de la ciencia de datos. Se identifican las preguntas de negocio que se intentarán responder con el proyecto de ciencia de datos.
- **Etapa 2 - Enfoque analítico:** En esta etapa se selecciona las técnicas de Cienca de Datos y Machine Learning que se utilizarán para responder las preguntas de negocio identificadas en la etapa anterior.

- **Etapa 3 - Requisitos de datos:** En esta etapa se identifican los datos necesarios para responder las preguntas de negocio.
- **Etapa 4 - Recopilación de datos:** En esta etapa se recopilan los datos necesarios segun lo identificado en la etapa anterior.
- **Etapa 5 - Comprensión de los datos:** En esta etapa se realiza un análisis exploratorio de los datos para comprenderlos mejor y detectar problemas de calidad de datos.
- **Etapa 6 - Preparación de los datos:** En esta etapa se preparan los datos para el entrenamiento de los modelos de Machine Learning. Esto incluye la limpieza de los datos, la selección de los datos relevantes y la transformación de los datos.
- **Etapa 7 - Modelado:** En esta etapa se entrenan los modelos de Machine Learning.
- **Etapa 8 - Evaluación:** En esta etapa se evalúa el rendimiento de los modelos de Machine Learning.
- **Etapa 9 - Implementación:** En esta etapa se despliegan los modelos de Machine Learning en producción.
- **Etapa 10 - Retroalimentación:** En esta etapa se monitorea el rendimiento de los modelos de Machine Learning en producción.

2.2 Repositorio institucional DSpace

DSpace es un software de código abierto que permite a las instituciones gestionar repositorios digitales. Es una solución flexible y escalable que puede utilizarse para almacenar una amplia variedad de contenidos digitales, como libros, tesis, artículos, datos de investigación, imágenes, audio y vídeo.

DSpace está organizado en tres niveles (ver Figura 3):

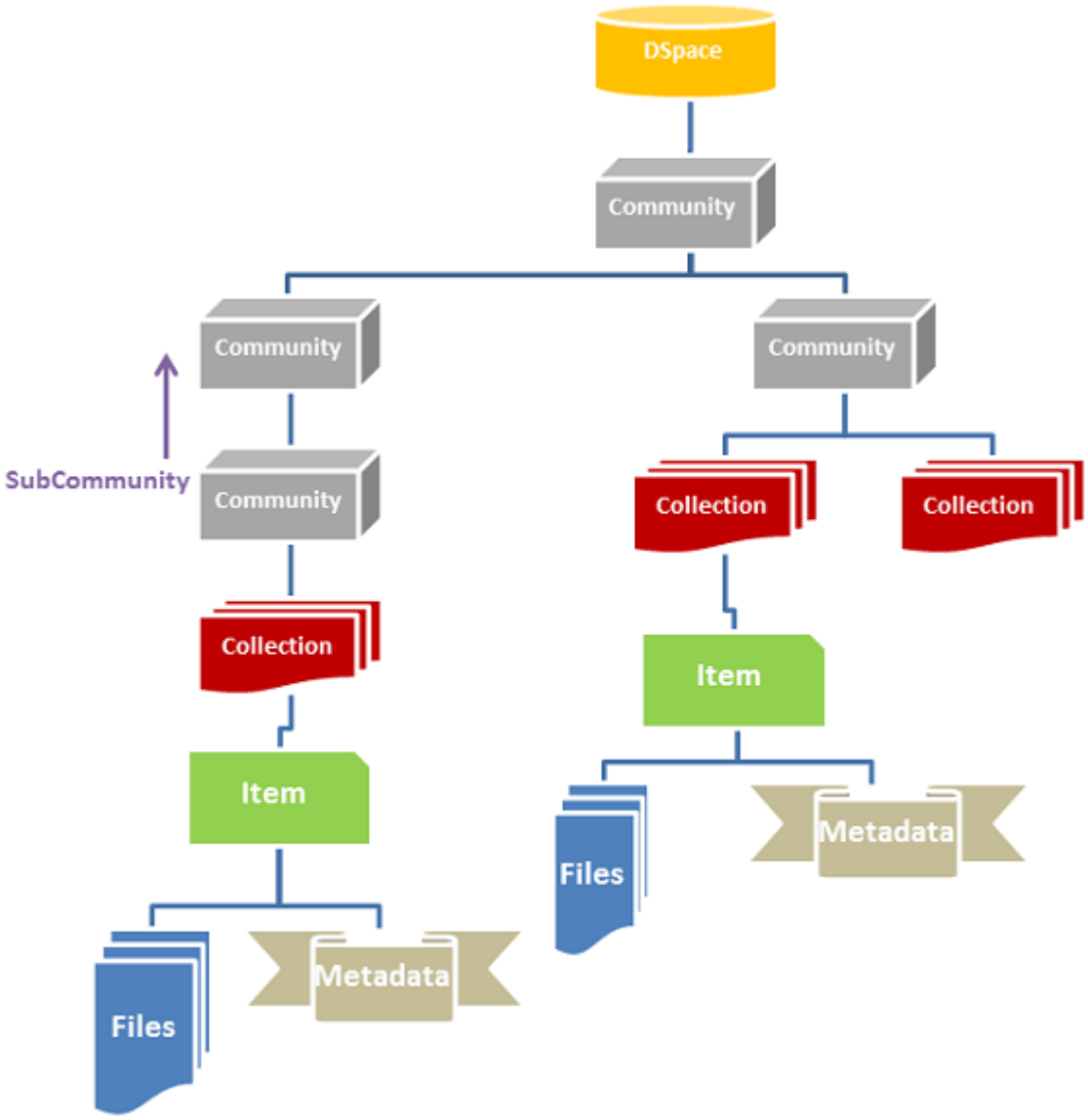
- Comunidades: Las comunidades son agrupaciones de usuarios que comparten un interés común. Por ejemplo, una universidad podría tener una comunidad para cada departamento académico o una empresa podría tener una comunidad para cada producto o servicio.
- Colecciones: Las colecciones son agrupaciones de ítems que comparten un tema o propósito común. Por ejemplo, una universidad podría tener una colección para cada disciplina académica

o una empresa podría tener una colección para cada tipo de archivo.

- Ítems: Los ítems son los elementos individuales que se almacenan en el repositorio. Pueden ser archivos de texto, imágenes, vídeos, audios, etc. Cada ítem tiene un conjunto de metadatos asociados que describen su contenido y origen. Los metadatos pueden utilizarse para buscar y recuperar ítems.

La organización de DSpace en comunidades, colecciones e ítems proporciona una estructura flexible que permite a las instituciones gestionar sus repositorios digitales de forma eficiente (DSpace, 2023).

Figura 3 - Estructura de DSpace.



Fuente: DSpace (2023).

2.2.1 Metadatos

Los metadatos pueden ser de dos tipos:

- **Metadatos descriptivos:** Estos metadatos proporcionan información sobre el contenido del ítem, como el título, el autor, la fecha de publicación, el tema, el formato, etc.
- **Metadatos administrativos:** Estos metadatos proporcionan información sobre el origen y el estado del ítem, como el nombre del creador, la fecha de creación, la versión, etc.

DSpace utiliza un conjunto de metadatos predeterminados, que se basan en el estándar DCMI Metadata Terms (DCT). Los DCT son un conjunto de términos y definiciones para describir recursos digitales.

Sin embargo, las instituciones pueden personalizar los metadatos de DSpace para satisfacer sus necesidades específicas. Por ejemplo, una universidad podría agregar metadatos para describir el departamento académico al que pertenece un ítem o el grado académico del autor.

Los metadatos son importantes para la gestión de repositorios digitales porque permiten a los usuarios buscar y recuperar ítems de forma eficiente. También pueden utilizarse para mejorar la visibilidad de los ítems y para facilitar su intercambio y reutilización.

2.2.2 Mecanismos de busqueda y recuperación de información en DSpace

DSpace ofrece una variedad de mecanismos de recuperación para ayudar a los usuarios a encontrar los ítems que buscan. Estos mecanismos incluyen:

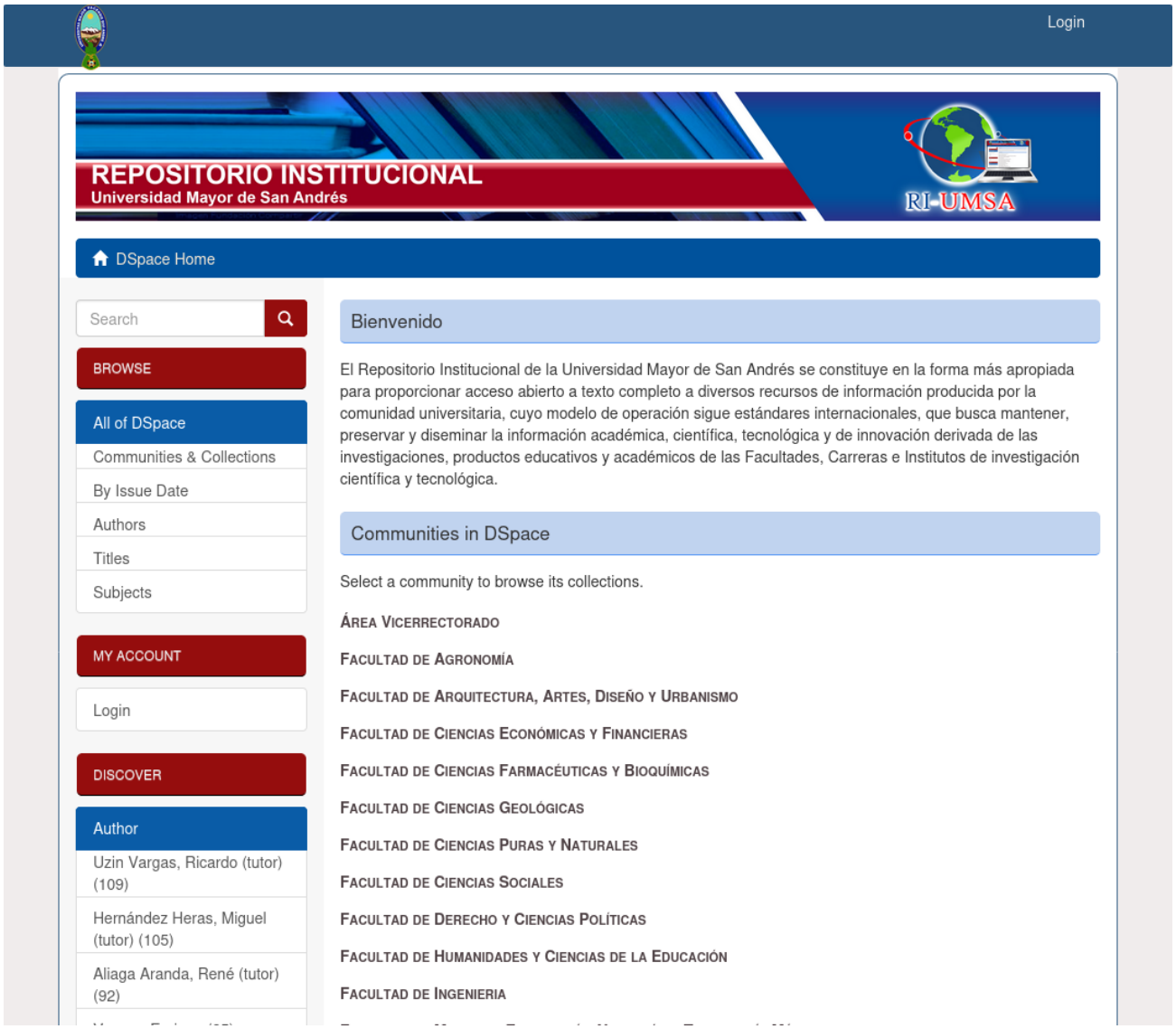
- **Búsqueda por texto completo:** Esta es la forma más básica de búsqueda, que permite a los usuarios buscar por palabras o frases que aparecen en el contenido de los ítems.
- **Búsqueda por metadatos:** Esta búsqueda permite a los usuarios buscar por valores de metadatos específicos, como el título, el autor, el tema o la fecha de publicación. Búsqueda por comunidades y colecciones: Esta búsqueda permite a los usuarios buscar por comunidades o colecciones específicas.
- **Búsqueda avanzada:** Esta búsqueda ofrece una variedad de opciones avanzadas, como la posibilidad de buscar por operadores booleanos, rangos de fechas y expresiones regulares.

2.2.3 Uso del repositorio DSpace por la UMSA

La Universidad Mayor de San Andrés utilizó el repositorio DSpace para catalogar documentos académicos, entre ellos memorias de proyectos de grado, tesis de grado y memorias laborales.

La figura Figura 4 muestra la página principal del repositorio.

Figura 4 - Pagina principal del repositorio DSpace implementado por la UMSA.



Fuente: Elaboración propia.

La UMSA utilizó la estructura de comunidades y colecciones de DSpace para organizar los documentos académicos. Cada comunidad representa una facultad y cada colección representa una carrera. Por ejemplo, la comunidad “Facultad de Ciencias Puras y Naturales” contiene las colecciones “Biología”, “Física”, “Matemáticas”, etc.

Como metadatos descriptivos, la UMSA utilizó los metadatos predeterminados de DSpace, como el título, el autor, etc. También agregó metadatos personalizados para describir el rol de cada contribuidor. Por ejemplo, la propiedad `dc.contributor.author` identifica al autor del documento y la propiedad `dc.contributor.advisor` identifica a los tutores. La figura Figura 5 muestra la vista individual de un ítem, que contiene los metadatos que lo describen y el archivo del documento para su descarga.

La tabla Tabla 1 enumera todas las propiedades utilizadas en el repositorio, junto con una descripción de su propósito.

Tabla 1 - Metadatos usados para describir a un item en el repositorio institucional de la UMSA

Nombre de la propiedad	Descripción
dc.contributor.advisor	Tutor del documento
dc.contributor.author	Autor del documento
dc.date.accesioned	Última fecha de consulta del documento
dc.date.available	Fecha desde la que el documento esta disponible en el repositorio
dc.date.issued	Fecha en la que el documento fue realizado
dc.identifier.url	URL del documento
dc.description.abstract	Resumen del documento
dc.language.iso	Siglas del lenguaje en el cual fue escrito el documento
dc.subject	Tópico del documento (también llamado descriptor)
dc.title	Título del documento
dc.type	Tipo del documento

Nombre de la propiedad	Descripción
dc.thesisdegreegrantor	Unidad que otorga el título obtenido como parte de la presentación del documento. Esta compuesta de el el nombre de la Universidad, el nombre de la facultad y el nombre de la carrera que lo otorga.
dc.thesisdegreename	Nombre del título que se otorga como parte de la presentación del documento.

Fuente: Elaboración propia.

Como formato estándar de publicación de los documentos, la UMSA eligió el formato PDF.

2.3 Procesamiento de Lenguaje Natural

Tal como lo define Amazon Web Services (s.f.), el Procesamiento del Lenguaje Natural (NLP) es una "tecnología de aprendizaje automático que otorga a las computadoras la capacidad de interpretar, manipular y comprender el lenguaje humano".

Las empresas utilizan el NLP para automatizar diversas tareas como:

- Clasificar correos no deseados.
- Traducir texto de un idioma a otro.
- Extraer información desde comentarios. Por ejemplo para procesar la retroalimentación de los clientes de un sitio e-commerce y extraer los puntos importantes de sus comentarios.
- Corregir la gramática de textos.
- Detectar plagio en documentos académicos.
- Búsqueda de documentos (al construir motores de búsqueda).
- Chatbots para brindar servicio al cliente de manera automatizada.
- Responder preguntas relacionadas con quién, qué, cuándo y dónde.
- Y muchas otras aplicaciones más.

Figura 5 - Repositorio de la UMSA, vista individual de ítem

Plan estratégico de marketing para posicionar a la Radio Universitaria San Andrés

dc.contributor.advisor	Valenti Morales, Carlos Eduardo, tutor	
dc.contributor.advisor	Gómez Lizarro, Israel Omar, tutor	
dc.contributor.author	Mamani Fernandez, Milka Belen	
dc.date.accessioned	2023-11-24T20:00:03Z	
dc.date.available	2023-11-24T20:00:03Z	
dc.date.issued	2023	
dc.identifier.uri	http://repositorio.umsa.bo/xmlui/handle/123456789/33845	
dc.description.abstract	La radio universitaria San Andrés es un medio de comunicación radiofónico institucional que se encuentra al servicio de la comunidad universitaria y la sociedad, dedicada al desarrollo de programas académicos, culturales, educativos y tecnológicos, con la finalidad de brindar conocimiento a los oyentes. Actualmente la radio no cuenta con un área de marketing que se especialice en visibilizar su marca o posicionarla dentro del mercado radiofónico actual lo que llevo a la radio a ser poco conocida dentro de la comunidad universitaria de la Universidad Mayor de San Andrés y la sociedad de la ciudad de La Paz, que como resultado llevo a la cancelación de varios programas por la baja audiencia por lo que se requiere de un plan de marketing que pueda identificar las necesidades de los oyentes para realizar programas con valor agregado. En este sentido el objetivo de presente trabajo fue Elaborar un Plan estratégico de Marketing para posicionar a la Radio Universitaria "San Andrés" en la comunidad estudiantil de la Universidad Mayor de San Andrés. La presente investigación demuestra el bajo conocimiento que tiene la comunidad universitaria sobre la existencia de la Radio Universitaria "San Andrés", de 398 universitarios encuestados solo el 31% conoce sobre su existencia, también se pudo evidenciar que los universitarios que más escuchan la radio son personas entre las edades de 18 a 30 años, en su mayoría prefieren programas que duren máximo una hora con temáticas enfocados en sus carreras, temas culturales o de difusión científica. A partir de ello se diseñó un Plan de Marketing, con propuestas enfocadas en los oyentes universitarios, con el objetivo de posicionar a la radio Universitaria San Andrés a través de estrategias de marketing mix.	es_ES
dc.language.iso	es	es_ES
dc.subject	PLAN ESTRATÉGICO DE MARKETING	es_ES
dc.subject	RADIO UNIVERSITARIA SAN ANDRÉS	es_ES
dc.subject	POSICIONAMIENTO	es_ES
dc.subject	ESTRATEGIAS DE MARKETING MIX	es_ES
dc.title	Plan estratégico de marketing para posicionar a la Radio Universitaria San Andrés	es_ES
dc.type	Thesis	es_ES
dc.thesisdegreegrantor	Universidad Mayor de San Andrés. Facultad de Ciencias Económicas y Financieras. Carrera de Administración de Empresas	es_ES
dc.thesisdegreename	Licenciado en Administración de Empresas	es_ES

Files in this item

Name: TD-3271.pdf
Size: 3.592Mb
Format: PDF

View/Open

This item appears in the following Collection(s)

- Trabajos Dirigidos

Fuente: Elaboración propia.

Para comprender el lenguaje humano, es necesario que las computadoras puedan transformar tanto el lenguaje escrito como hablado en representaciones numéricas. Estas representaciones numéricas se utilizan como entrada para algoritmos computacionales, permitiendo al sistema comprender instrucciones, responder preguntas u realizar otras tareas como respuesta.

En el procesamiento del lenguaje natural, es común que los sistemas estén compuestos por una serie de componentes interconectados, formando lo que se conoce como un Pipeline. Este Pipeline, por lo general, consta de uno o más de los siguientes elementos:

- **Tokenizer:** Encargado de segmentar el texto para hacerlo mas manejable computacionalmente. La tokenización puede realizarse de diversas formas, como la separación del texto en palabras, convirtiendo cada palabra en un segmento o *token*.
- **Stopwords Remover:** Elimina *tokens* comunes que no aportan un significado esencial al texto, como “el”, “es”, “y”, entre otros.
- **Stemming/Lematización:** Reduce los *tokens* a su forma raíz, por ejemplo, convirtiendo “corriendo” a su forma raíz “correr”.
- **Word Embeddings:** Convierte los tokens en vectores densos de baja dimensión que capturan su significado semántico.

Una vez obtenidas las representaciones numéricas de los *tokens*, otros componentes se encargan de tareas más especializadas, como:

- **Part-of-Speech (POS) tagging:** Asigna a cada *token* categorías gramaticales. Por ejemplo, en el texto “José estudia”, categoriza o *etiqueta* “José” como “PROPN” (sustantivo propio) y “estudia” como “VERB” (verbo).
- **Dependency Parsing:** Analizar la estructura gramatical y la relación entre las palabras en una secuencia de texto. Por ejemplo al identificar, en un oración, qué adjetivos están describiendo a qué sustantivos. Claramente se ve la necesidad de crear un árbol de relaciones o dependencias gramaticales. Construir este árbol es la función de este componente.
- **Named-entity Recognition (NER):** Identifica y clasifica entidades en un texto dado, entidades como ser nombres, organizaciones, lugares, etc.

Estas áreas han sido objeto de estudio durante décadas, y su implementación ha variado a lo largo del

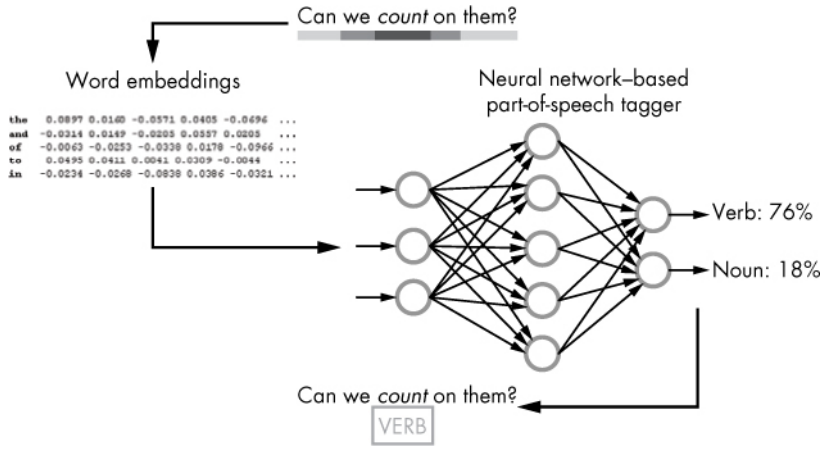
tiempo, con resultados cada vez más precisos. Las implementaciones iniciales se basaban en reglas predefinidas, pero debido a la ambigüedad inherente del lenguaje natural, no lograban alta precisión. Con la llegada del Machine Learning y del Deep Learning (ver Figura 7), los sistemas de NLP han evolucionado significativamente, permitiendo ya no programar reglas de manera explícita sino mas bien entrenar redes neuronales artificiales para llevar a cabo estas tareas y otras nuevas como:

- **Análisis semántico:** Determinar el sentimiento en un texto. Por ejemplo clasificar si un comentario es positivo, negativo o neutral.
- **Topic Modeling:** Identificar tópicos o temas tratados en una larga colección de documentos.
- **Similaridad Semántica:** medir la similitud en significado entre dos textos.
- **Clasificación de texto:** Categorizar texto en predefinidas categorías o clases. Por ejemplo, en una aplicación de chat de soporte clasificar si el usuario esta pidiendo información, o esta haciendo un reclamo, o esta solicitando soporte técnico. En este ejemplo las categorías a clasificar podrían ser: “solicitud de información”, “reclamo” y “soporte”.
- **Seq2seq:** Generar una secuencia de texto basado en una secuencia de texto. Ejemplos de esta tarea son: la traducción de texto de un idioma a otro, el resumen de textos, la respuesta a preguntas.
- **Generación de texto:** Generar texto basado en patrones aprendidos.

Especialmente las dos últimas tareas mencionadas se han visto favorecidas por el desarrollo de Modelos de Lenguaje Grande (LLMs) los cuales se caracterizan por poseer un elevado número de parámetros entrenables, que a menudo alcanzan la escala de billones.

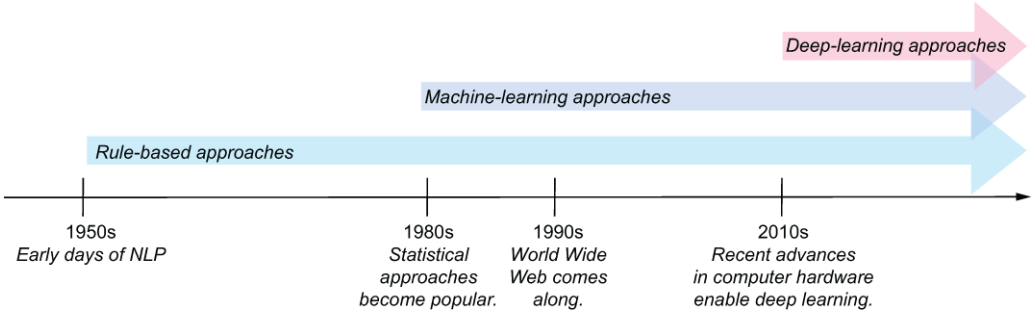
En la Figura 6 se muestra el diagrama general de un sistema de Part-of-speech tagging desarrollado en base a un modelo ML. En este se usa una red neuronal como modelo, el cual aprende a predecir la categoría gramatical a la que pertenece cada una de las palabras del texto de entrada.

Figura 6 - Part-of-speech tagging implementado con una red neuronal.



Fuente: Vajjala et al. (2020).

Figura 7 - Linea de tiempo del NLP mostrando las diferentes estrategias usadas.



Fuente: Kochmar (2022), sección 1.1.

El entrenamiento de un modelo de ML para resolver una tarea de NLP es similar al entrenamiento convencional de cualquier modelo de ML y generalmente implica:

1. **Representar el Texto:**

   El texto se convierte en una forma que los modelos de ML puedan entender. Esto implica la tokenización, donde el texto se divide en unidades más pequeñas, como palabras, sub-palabras o caracteres. Luego, estas unidades se representan en forma numérica para que los modelos puedan trabajar con ellas.

2. **Seleccionar el Modelo:**

   Hay varios tipos de modelos de ML utilizados en NLP, como:

1. **Bag-of-Words (BoW) y Modelos N-gram:** Representan el texto como vectores de conteo de palabras o secuencias de palabras adyacentes.
2. **Word Embeddings:** Transforman palabras en vectores de números reales que capturan relaciones semánticas y contextuales (Word2Vec, GloVe, FastText).
3. **Redes Neuronales Recurrentes (RNN):** Modelos que procesan secuencias de texto y pueden recordar información anterior.
4. **Redes Neuronales Convolucionales (CNN):** Útiles para tareas como clasificación de texto, extracción de características.
5. **Transformers:** Modelos basados en atención que han mostrado un excelente desempeño en varias tareas de NLP (BERT, GPT, etc.).

3. **Entrenar el Modelo:**

   El modelo seleccionado se entrena con conjuntos de datos etiquetados. Durante el entrenamiento, el modelo ajusta sus pesos o parámetros para aprender patrones y relaciones entre las representaciones numéricas de entrada y las salidas esperadas.

4. **Evaluar el Modelo:**

   Una vez entrenado, se evalúa el modelo utilizando conjuntos de datos separados que no se usaron durante el entrenamiento (conjunto de validación o prueba). Se utilizan métricas específicas de la tarea (precisión, recall, F1-score, etc.) para medir su rendimiento.

Una vez que el modelo ha sido entrenado y evaluado satisfactoriamente, se puede utilizar para realizar predicciones en nuevos datos o situaciones del mundo real. Esto puede incluir tareas como clasificación de texto, generación de texto, traducción automática, análisis de sentimientos, entre otras.

2.3.1 Named-entity Recognition

El Reconocimiento de Entidades Nombradas (NER por sus siglas en inglés) se refiere a la tarea NLP de identificar *entidades* en un texto. Por lo general, las entidades son nombres de personas, ubicaciones y organizaciones, y otros tipos de texto especiales, como expresiones monetarias, fechas, nombres de productos, nombres o números de leyes o artículos, y similares (Vajjala et al. (2020), capítulo 5, sección Named Entity Recognition).

Como lo muestra la Figura 8, dado un texto, se espera que un sistema NER identifique los nombres de personas, lugares, y otras entidades.

Figura 8 - Ejemplo de un sistema NER.

Fuente: Elaboración propia.

Los tipos de entidades a identificar por un sistema NER son definidos por el desarrollador del sistema. Por ejemplo, un sistema NER para extraer datos bibliográficos de documentos académicos podría identificar entidades como ser: nombres de autores, títulos de documentos, nombres de revistas, etc.

2.3.1.1 Construyendo un sistema NER

Dos formas de construir un sistema NER son descritas en las siguientes secciones. Primeramente se describe como desarrollar un sistema basado en reglas, en el que se programan los patrones que

las entidades a reconocer siguen. Seguidamente se describe como desarrollar un sistema basado en Machine Learning, en el que no se programan explícitamente los patrones, sino mas bien el sistema *aprende* a reconocerlos.

2.3.1.1.1 Ner basado en reglas

Este sistema esta basado en una lista compilada de patrones. Los patrones pueden describir el tipo de texto a buscar, por ejemplo decir “encontrar un texto que este compuesta de dos palabras, la primera es un sustantivo propio y la segunda palabra es el texto literal ‘nació’ ”. Asi pues los texto encontrado por este patrón podrían ser “Julio Cesar nació”, “Simon Bolivar nació”, etc.

Estas reglas pueden programarse para cubrir tantos casos como sea posible y construir un sistema NER basado en reglas. Herramientas como RegexNER de Stanford NLP y EntityRuler de spaCy brindan funcionalidades para implementar un propio Modelo NER basado en reglas.

2.3.1.1.2 Ner basado en Machine Learning

Un enfoque más práctico para el NER es entrenar un modelo de ML, el cual puede predecir las entidades en un texto no visto previamente. Para cada palabra del texto, el modelo tomará una decisión sobre si esa palabra es o no una entidad y, si lo es, qué tipo de entidad es.

Tradicionalmente, el NER basado en ML se modela como un problema de clasificación de secuencias, en la que la secuencia esta compuesta de porciones de texto, llamados tokens, donde cada token es clasificado en una de N categorías (los tipos de entidades).

La clasificación de cada token puede ser llevada considerando o no el contexto que rodea al token. Sin contexto en el caso de que cada token sea clasificado sin tomar el cuenta en el contexto en el que el token es usado. Con contexto en el caso de que si se consideren los token vecinos. Generalmente se prefiere implementar el sistema tomando el contexto ya que el significado de un token suele depender de las palabras circundantes. Por ejemplo el texto “Washington” podría ser identificado como una entidad LOC (lugar, refiriéndose a un estado de E.E.U.U.), pero si se toman en cuenta el texto circundante “George Washington” se sabe que el texto debió ser etiquetado como PER (nombre de persona).

Para lograr capturar el contexto existen diferentes estrategias, una de ellas es usar una arquitectura de ML como las Redes Neuronales Convolucionales.

2.3.1.2 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNNs por sus siglas en inglés) son ampliamente conocidas por su uso en la visión por computadora, pero también se han aplicado con éxito en tareas de Procesamiento del Lenguaje Natural (NLP) para capturar el contexto en texto. A pesar de que las CNNs fueron inicialmente diseñadas para trabajar con datos de imágenes, se pueden adaptar para procesar datos secuenciales, como el texto, mediante el uso de operaciones de convolución.

Aquí hay algunos enfoques comunes sobre cómo las CNNs capturan el contexto en tareas de NLP:

1. **Operaciones de convolución en texto:** Las CNNs aplican operaciones de convolución a través de ventanas de palabras (también conocidas como filtros) sobre secuencias de palabras o caracteres. Estas operaciones buscan patrones locales, detectando combinaciones específicas de palabras o caracteres que podrían ser características relevantes para comprender el contexto. En la Figura 9 se muestra un ejemplo del uso de ventanas de contexto las cuales son luego procesadas por una CNN.

2. **Extracción de características:** Las capas convolucionales de una CNN pueden actuar como extractores de características al aplicar múltiples filtros de diferentes tamaños. Cada filtro se desliza a través de la secuencia de palabras para detectar patrones locales, capturando información sobre la estructura del texto en diferentes escalas.

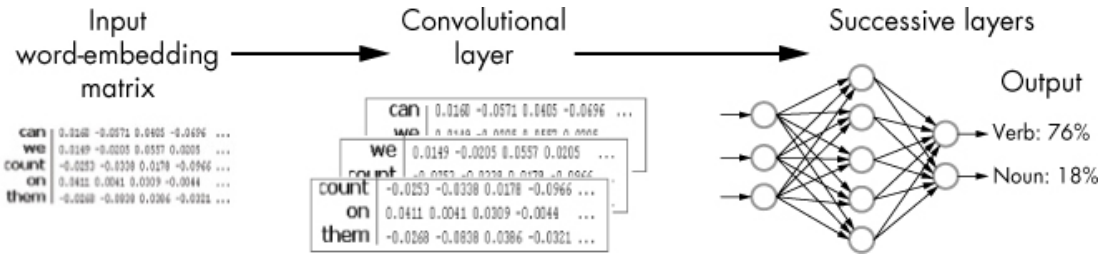
3. **Capas de agrupamiento (pooling):** Después de aplicar las operaciones de convolución, se utilizan capas de agrupamiento, como las capas de max-pooling, para reducir la dimensionalidad y resaltar las características más relevantes. Estas capas identifican las características más importantes extraídas por las convoluciones, conservando la información más relevante para el contexto.

4. **Transferencia de aprendizaje:** Las CNNs pre-entrenadas en grandes conjuntos de datos pueden ser utilizadas en NLP como extractores de características. Estas redes pre-entrenadas pueden transferir su capacidad para capturar patrones y contextos relevantes a tareas específicas de NLP a través de técnicas como el ajuste fino (fine-tuning).

2.3.1.3 Etiquetas NER

Las etiquetas NER son simplemente la etiqueta que un sistema NER asigna a cada token o palabra de

Figura 9 - Mirada de como una CNN funciona para tareas de NLP.



Fuente: Vajjala et al. (2020).

un texto dado. Por ejemplo en la Figura 8, puede decirse que a las palabras “Ing. Eduardo Garay” el sistema NER les ha asignado la etiqueta PER.

2.3.2 Extracción de palabras clave

La extracción de palabras clave en NLP es el proceso de identificar automáticamente las palabras o frases más importantes en un documento de texto o corpus. Estas palabras clave sirven como componentes cruciales para varias tareas de NLP, incluyendo:

- **Recuperación de información:** Las palabras clave extraídas se pueden utilizar para buscar e indexar documentos, facilitando la búsqueda de información relevante.
- **Resumen de documentos:** Las palabras clave pueden ayudar a condensar los puntos principales de un documento en un resumen conciso.
- **Clasificación de texto:** Las palabras clave extraídas se pueden utilizar para categorizar documentos en diferentes categorías según su contenido.

¿Cómo funciona?

Existen dos enfoques principales para la extracción de palabras clave:

- **Extractivo:** Este método selecciona directamente palabras o frases importantes del texto en base a varios criterios. Algunas técnicas comunes incluyen:
  - **Frecuencia:** Las palabras clave que aparecen más frecuentemente se consideran más importantes.
  - **TF-IDF (Frecuencia del Término-Frecuencia Inversa del Documento):** Este método equilibra la frecuencia dentro de un documento con la rareza en un corpus más amplio, favoreciendo palabras clave específicas e informativas.

  - **Basado en la posición:** Las palabras que aparecen al principio de las oraciones o párrafos a menudo se priorizan.
  - **Métodos estadísticos:** Técnicas como el análisis de n-gramas o la coocurrencia de frases clave pueden identificar grupos de palabras clave estadísticamente significativos.

- **Abstractivo:** Este método va más allá de la superficie y genera nuevas palabras clave que capturan la esencia del texto, incluso si no están presentes explícitamente. Esto se puede lograr mediante:
  - **Modelos de aprendizaje profundo:** Algoritmos avanzados como RNN, Transformers y autoencoders pueden analizar el contexto del texto y generar palabras clave relevantes.
  - **Etiquetado de Categorías Gramaticales:** Identificar la Categoríá Gramatical de las palabras puede revelar conceptos y entidades importantes que se pueden usar como palabras clave.

Elegir el enfoque correcto:

El mejor método para la extracción de palabras clave depende de sus necesidades específicas y la naturaleza de sus datos de texto. Los métodos extractivos son generalmente más simples y rápidos, pero pueden no capturar conceptos complejos. Los métodos abstractivos ofrecen una comprensión más profunda, pero pueden ser computacionalmente más caros y menos interpretables. Los enfoques híbridos que combinan ambos métodos también pueden ser efectivos.

2.4 Spacy

SpaCy es una biblioteca gratuita y de código abierto para el procesamiento del lenguaje natural (NLP) en Python. Está diseñada para ser rápida, eficiente y escalable ((ExplosionAI, s.f.-a)).

SpaCy proporciona una serie de funciones para NLP, que incluyen:

- **Tokenización:** SpaCy puede tokenizar el texto en palabras, frases y otros tokens.
- **Etiquetado de categorías gramaticales:** SpaCy puede identificar la categoría gramatical de cada token (por ejemplo, sustantivo, verbo, adjetivo, etc.).
- **Reconocimiento de entidades nombradas:** SpaCy puede identificar entidades nombradas, como personas, lugares y organizaciones.

- **Análisis sintáctico de dependencias:** SpaCy puede identificar las relaciones entre las palabras en una oración.
- **Similaridad semántica:** SpaCy puede medir la similitud entre dos piezas de texto.

Algunas de sus características son:

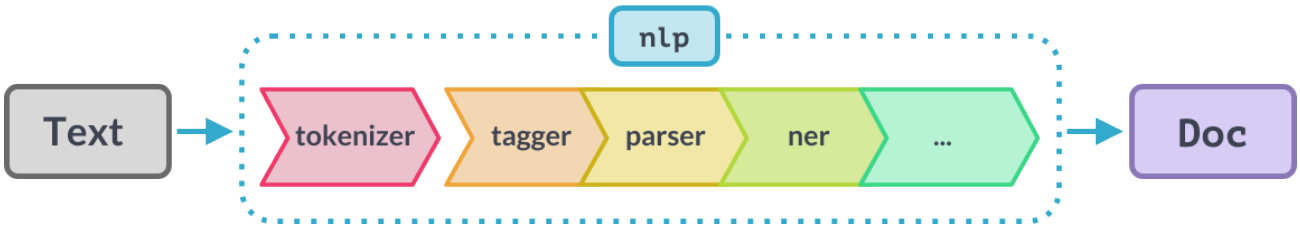
- Soporte para 74 idiomas
- Word Vectors pre-entrenados
- Velocidad State-of-the-art (SOTA)
- Sistema de entrenamiento de modelos ML listo para producción
- Tokenización lingüísticamente motivada
- Componentes para NER, POS Tagging, Dependency Parsing, Sentence Segmentation, Clasificación de Texto, Lematización, Análisis Morfológico, Entity Linking y más.
- Fácilmente extensible con componentes personalizados
- Soporte para modelos personalizados en PyTorch, TensorFlow y otros frameworks
- Visualizadores incluidos para NER
- Fácil empaquetamiento de modelos, despliegue a producción y administración del flujo de trabajo
- Robusta y rigurosamente evaluada precisión

2.4.1 Pipelines en spaCy

En SpaCy, una pipeline es una secuencia de componentes que procesan texto en un orden específico. Cada componente realiza una tarea diferente, como Tokenización, Etiquetado de Categorías Gramaticales, Reconocimiento de Entidades Nombradas o Análisis Sintáctico de Dependencias. La salida de un componente se pasa como entrada al siguiente componente, lo que permite un enfoque en capas del análisis de texto.

Por ejemplo, uno de los pipelines pre-construidos que vienen junto a la librería es el `es_core_news_lg` el cual esta constituido por la cadena de componentes mostrada en la Figura 10.

Figura 10 - NLP pipeline del modelo `es_core_news_lg` de spaCy.



Fuente: spaCy (s.f.).

Los componentes de la pipeline se ejecutan en el siguiente orden:

- **Tokenizer:** Divide el texto en tokens (palabras, puntuación, etc.).
- **Tagger:** Asigna etiquetas de Categorías Gramaticales a cada token (por ejemplo, sustantivo, verbo, adjetivo).
- **Lemmatizer:** Reduce las palabras a sus formas base (lemas).
- **Parser:** Identifica la estructura sintáctica de la oración y las relaciones entre palabras.
- **Entity recognizer:** Encuentra y clasifica entidades nombradas (por ejemplo, personas, organizaciones, lugares).
- **Componentes personalizados:** También puede agregar componentes personalizados a la pipeline para realizar tareas adicionales.

También, es posible construir un pipeline personalizado que incluya solo los componentes necesarios para una tarea NLP. Por ejemplo, un pipeline para la tarea de NER podría incluir solo los componentes de Tokenizer y NER. En este caso, el texto se pasa primero al componente Tokenizer, que divide el texto en tokens. Luego, los tokens se pasan al componente NER, que identifica las entidades nombradas en el texto.

2.4.2 Pipeline de NER basado en Entity Ruler

En SpaCy, un `EntityRuler` es un componente poderoso usado para crear un sistema de *Reconocimiento de Entidades Nombradas basado en reglas*. Permite definir patrones personalizados para identificar y clasificar entidades específicas en un texto, complementando o incluso reemplazando los modelos estadísticos de NER incluidos en las pipelines de SpaCy.

Aquí hay un desglose de que hace un componente EntityRuler:

1. Definir patrones:

Se puede definir patrones para diferentes tipos de entidades usando diccionarios. Estos patrones pueden basarse en:

- **Propiedades de los tokens:** Como texto en minúsculas, etiquetas de categorias gramaticales o palabras específicas.
- **Frases:** Coincidencia con secuencias exactas de palabras o tokens.
- **Expresiones regulares:** Patrones más complejos para un reconocimiento de entidades flexible.

2. Asignar etiquetas:

Cada patrón está asociado con una **etiqueta** que identifica el tipo de entidad que tiene por objetivo. Algunas etiquetas comunes incluyen `PERSONA`, `ORG`, `LOC`, `FECHA`, etc., pero se puede crear cualquier etiqueta personalizada relevante para una tarea dada.

3. Procesar texto:

Cuando se aplica el EntityRuler a un documento de texto, este busca coincidencias de dentro de los tokens en base a los patrones definidos. Si el patrón es encontrado, la etiqueta correspondiente se asigna al espacio de texto correspondiente, creando una nueva entidad nombrada.

4. Integración con pipelines:

El EntityRuler se puede integrar fácilmente en un Pipeline de SpaCy. Puedes ser agregado antes o después del componente NER existente, dependiendo del flujo de trabajo deseado.

Ventajas de usar un EntityRuler:

- **Alta personalización:** Permite definir patrones específicos para capturar entidades que un NER estadístico podría pasar por alto.
- **Extracción dirigida:** Se enfoca en reconocer solo las entidades que son relevantes para una tarea.
- **Mayor precisión:** Puede mejorar el rendimiento general de NER, especialmente para entidades raras o poco convencionales.

Limitaciones de usar un EntityRuler:

- **Mantenimiento de reglas:** Requiere esfuerzo manual el definir y mantener patrones, lo que

puede llevar mucho tiempo.

- **Generalización limitada:** Las reglas pueden no funcionar bien en datos no vistos o dominios diferentes.
- **Posible sobreajuste:** Patrones demasiado específicos pueden llevar a un bajo Recall y a no identificar entidades relevantes.

Para ejemplificar la creación de un pipeline NER basado en Entity Ruler, se muestra un ejemplo de uso del Entity Ruler en la Figura 11. En el mismo se define un patrón para detectar nombres de Facultades de la UMSA. El patrón a detectar es "cualquier secuencia de texto que comience con el token 'facultad', seguido de 'de', seguido de un token alfanumérico y seguido opcionalmente de un otro token alfanumérico. El patrón entonces es añadido al Entity Ruler. Finalmente se muestra como el Entity Ruler clasifica los tokens de dos textos de ejemplo.

Se ve que para el segundo texto el Entity Ruler no clasifico como entidad las últimas dos palabras ("Y NATURALES"), que se sabe son parte del nombre de una facultad. Aquello es debido a que el patrón solo capturaba hasta dos tokens alfanuméricos después del token "de", y cualquier otro token posterior es simplemente ignorado. Esto demuestra que al definir reglas se necesita conocer todos los posibles patrones, y en que en algún punto puede hacerse imposible definir todas las reglas necesarias para obtener un buen rendimiento en la tarea. Para aquellos casos en los que definir reglas es inviable, lo conveniente es usar un modelos estadísticos, o modelos basados en Machine Learning que puedan aprender a reconocer patrones sin programarlos explícitamente. En la siguiente sección se describe en términos generales un pipeline NER basado en ML.

El Entity Ruler y su funcionamiento es explicado en más detalle en la documentación oficial[4].

2.4.3 Pipeline de NER basado en ML

Este pipeline esta compuesto de dos componentes, el Tokenizer y el componente NER basado en ML (ver Figura 12).

El componente *Tokenizer* es el mismo descrito en la Sección 2.4.2.

El componente NER, en este caso, esta basado en modelos de Machine Learning.

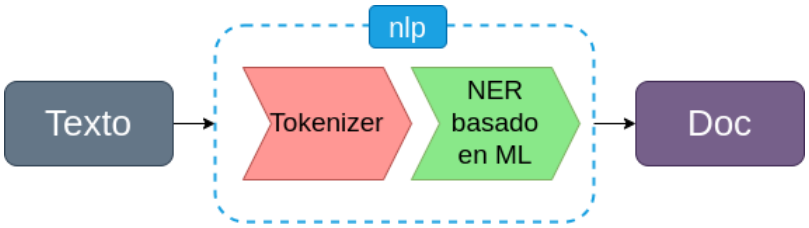
[4] https://spacy.io/usage/rule-based-matching#entityruler

Figura 11 - Ejemplo de uso del Entity Ruler

```
import spacy.lang.es import Spanish

# Inicialización del pipeline
nlp = Spanish()
# Agregar el Entity Ruler al pipeline
ruler = nlp.add_pipe("entity_ruler")

# Definir patrón para reconocer nombres de facultades
patterns=[{
    "label": "FACULTY",
    "pattern": [
        {"LOWER": "facultad"},
        {"LOWER": "de"},
        {"IS_ALPHA": True}, # cualquier palabra
        {"IS_ALPHA": True, "OP": "?"} # (opcional) cualquier palabra
    ]
}]
# Agregar patrón al Entity Ruler
ruler.add_patterns(patterns)

# Definir textos de ejemplo
samples=["Facultad de Ingeniería", "FACULTAD DE CIENCIAS PURAS Y NATURALES"]
# Procesar textos de ejemplo con el pipeline NER basado en Entity Ruler
# y mostrar entidades encontradas
for sample in samples:
    doc = nlp(sample)
    print([(ent.text, ent.label_) for ent in doc.ents])

# Output:
# ('Facultad', 'FACULTY')
# ('de', 'FACULTY')
# ('Ingeniería', 'FACULTY')
# ('FACULTAD', 'FACULTY')
# ('DE', 'FACULTY')
# ('CIENCIAS', 'FACULTY')
# ('PURAS', 'FACULTY')
```

Fuentes: Elaboración propia.

Figura 12 - Pipeline NLP especializado para Named-entity Recognition (NER)



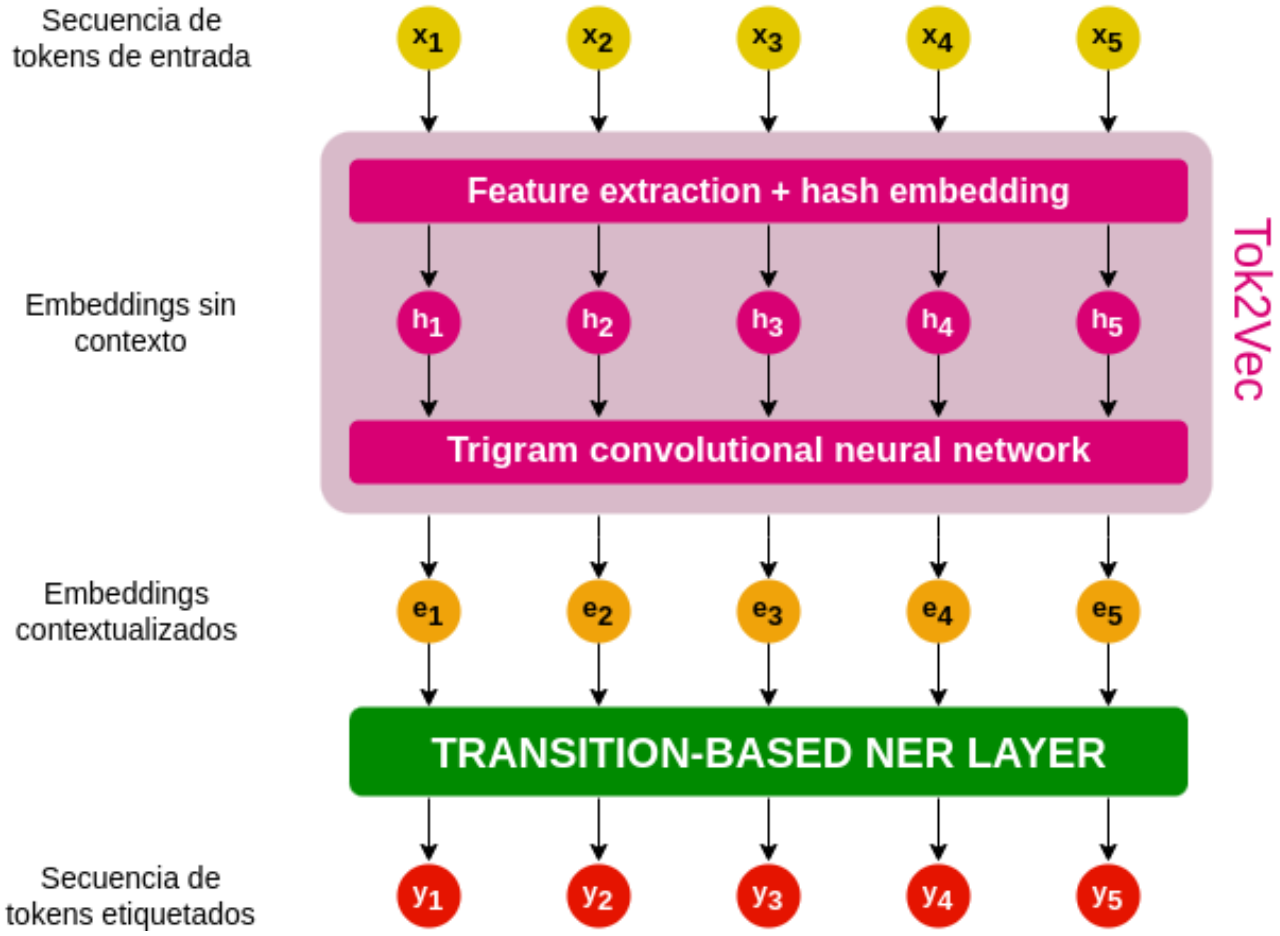
Fuente: Elaboración propia.

Existen varias arquitecturas de modelos ML predefinidos en la librería, una de las arquitecturas predefinidas es la `spacy.TransitionBasedParser.v2`. Se trata de un modelo con dos módulos: un módulo de Embeddings llamado `Tok2Vec` y un segundo módulo de clasificación o etiquetado NER. La Figura 13 muestra la estructura general de la arquitectura, y a continuación se describe a grandes rasgos la función de cada capa:

- **Tok2Vec:** conjunto de capas encargadas de obtener una representación vectorial, o *embedding* de cada *token* de entrada. La obtención de los *embeddings* se lo realiza en 2 etapas:
  - **Feature extraction + Hash Embedding:** En esta etapa se traduce cada *token* a su representación vectorial, o *hash embedding*, de acuerdo a un diccionario interno. En esta operación, se procesa cada *token* individualmente por lo que el resto de los *tokens* no influyen en el vector resultante, por ello se llama *embeddings sin contexto*.
  - **Trigram Convolutional Neural Network:** Traduce cada *embedding sin contexto* a una representación matricial donde los *tokens* vecinos son parte del resultado. De esta manera se crea una **ventana de contexto**. Esta ventana de contexto ayuda al modelo a tomar en cuenta las palabras que rodean una cierta palabra para determinar si una palabra es una entidad o no, por ejemplo, en caso de reconocimientos de nombres propios, dado un nombre "Juan" es muy probable que la siguiente palabra también sea parte del nombre , ya que los nombres suelen estar compuestos por nombre y apellido. Como parte de esta capa, la matriz resultante es nuevamente compactada a un vector, un vector que guarda el *contexto* ya que fue obtenida a partir de la conjunción del token actual junto a sus *tokens* vecinos, es por ello que spaCy los llama *embeddings con contexto*.
- **Transition-based layer:** Esta capa es la parte central del modelo. Consiste en una maquina de estados en la que se procesa todos los *embeddings contextualizados* en secuencia, donde en cada ciclo se determina una acción a realizar. Acciones posibles son, clasificar el *embedding* actual como una entidad X, clasificarlo como una entidad Y, o ignorarlo. La acción es determinada en base al estado actual de la maquina de estados el cual esta dado por los *embeddings* por clasificar y los ya clasificados. El entrenamiento de este modelo se trata en esencia de enseñar al modelo a que decisión tomar dado un estado.

La arquitectura que acaba de ser descrita es explicada en detalle por ExplosionAI (2017).

Figura 13 - Arquitectura del modelo NER predefinido `spaCy.TransitionBasedParser.v2`.



Fuente: Adaptado de Guillarme y Thuiller (2021).

2.4.3.1 Entrenamiento de un pipeline NER basado en ML en spaCy

El entrenamiento de un modelo NER en spaCy sigue el procedimiento tradicional para entrenar modelos de ML, esto es:

1. **Diseñar una arquitectura de modelo de ML**, o usar una arquitectura pre-definida que viene con la librería. Lo recomendado, es probar resolver la tarea con una de las arquitecturas predefinidas, y si fuese necesario utilizar una personalizada. El cómo usar modelos personalizado esta documentado en la página oficial de spaCy[5].

2. **Preparar un conjunto de datos de entrenamiento para entrenar el modelo.**

SpaCy espera que los datos de entrenamiento estén en un formato específico propio de la librería.

[5] https://spacy.io/usage/layers-architectures#frameworks

Cada dato de entrenamiento debe ser convertido en un objeto `Doc` (ver 2.4.4). El objeto `Doc` debe contener el texto de entrada, y las anotaciones NER. La Figura 15 muestra un ejemplo de un objeto `Doc` con las anotaciones NER añadidas. Luego, los objetos `Doc` deben ser agregados a un objeto `DocBin` y finalmente este archivo `DocBin` debe ser exportado como un archivo `.spacy` (ExplosionAI (s.f.-d))

Los datos de entrenamiento consisten en textos anotados con entidades, es decir textos donde cada *token* está etiquetado con el tipo de entidad con el que se lo identifica. Así pues, en la Figura 14 se muestra un ejemplo de un texto divido en tokens en donde a cada token se le ha asignado una etiqueta de entidad. Este ejemplo concreto sigue el estándar de etiquetado IOB.

El etiquetado IOB asigna la etiqueta `O` a los tokens que no forman parte de un entidad, y las etiquetas `B-<tipo de entidad>` y `I-<tipo de entidad>` a los tokens que sí. La `B-` indica que el token da comienzo a la entidad, mientras que `I-` indica que el token se encuentra al medio o al final de la entidad. Así, en el ejemplo, la entidad *AUTHOR* comienza con el token *"RODRIGO"*, continúa con el token *"ITURRALDE"*, y finaliza con el token *"ALIAGA"*.

spaCy internamente sigue el estándar IOB de anotaciones de etiquetas NER, sin embargo los datos de entrenamiento siguen otra notación. Las anotaciones deben añadirse a un objeto `Doc` (ver 2.4.4). La Figura 15 muestra un objecto `Doc` con las anotaciones NER añadidas. En la misma se puede apreciar que en vez de usar etiquetas IOB, spaCy describe las etiquetas NER mediante rangos de caracteres. Por ejemplo, en la misma figura, la ubicación de la entidad AUTHOR es descrita por `start_char` y `start_end` cuyos valores son 257 y 286, lo que indica que una entidad tipo AUTHOR esta ubicada entre los caracteres 257 y 286 del texto de entrada.

Luego de haber construido un conjunto amplio de datos de entrenamiento en forma de objetos `Doc`, lo siguiente es guardarlos agregarlos en un archivo `DocBin` y finalmente exportarlos como archivos `.spacy`.

Los archivos `.spacy` son entonces usados para el entrenamiento mediante los pasos descritos en la documentación oficial de spaCy[6].

[6] https://spacy.io/usage/training

Figura 14 - Ejemplo de un texto de entrada anotado con etiquetas IOB

UNIVERSIDAD	O
MAYOR	O
DE	O
SAN	O
ANDRES	O
\n	O
FACULTAD	B-DEPARTMENT
DE	I-DEPARTMENT
INGENIERIA	I-DEPARTMENT
\n	O
\n	O
PRESENTADO	O
POR	O
:	O
RODRIGO	B-AUTHOR
ITURRALDE	I-AUTHOR
ALIAGA	I-AUTHOR
\n	O
TUTOR	O
:	O
OSCAR	B-ADVISOR
MARCELO	I-ADVISOR
POMA	I-ADVISOR
CALLE	I-ADVISOR
AÑO	O
:	O
2023	B-YEAR

Fuente: Elaboración propia.

Figura 15 - Ejemplo del texto de una carátula anotada en el formato esperado por spaCy.


```
{
    "text": "UNIVERSIDAD MAYOR DE SAN ANDRÉS\nFACULTAD DE
    INGENIERÍA\nINGENIERÍA PETROLERA\n\nOPTIMIZACIÓN DE LA
    PRODUCCIÓN DE PETRÓLEO, MEDIANTE EL MÉTODO\nDE BOMBEO
    HIDRÁULICO TIPO JET \"BOMBA JET CONVENCIONAL\"\nAPLICADO EN
    EL POZO PATUJUSAL 8ST\n\nPOSTULANTE: Univ. ABIGAIL VALERIA
    APAZA CONDORI\nTUTOR: M. Sc. ING. SERGIO EDDY VISCARRA
    ORTUÑO\n\nLA PAZ - BOLIVIA\n2020",
    "ents": [
        {
            "start_char": 33,
            "end_char": 55,
            "label_": "FACULTY",
            "text": "FACULTAD DE INGENIERÍA"
        },
        {
            "start_char": 57,
            "end_char": 77,
            "label_": "DEPARTMENT",
            "text": "INGENIERÍA PETROLERA"
        },
        {
            "start_char": 81,
            "end_char": 235,
            "label_": "TITLE",
            "text": "OPTIMIZACIÓN DE LA PRODUCCIÓN DE PETRÓLEO,
            MEDIANTE EL MÉTODO\nDE BOMBEO HIDRÁULICO TIPO JET
            \"BOMBA JET CONVENCIONAL\"\nAPLICADO EN EL POZO
            PATUJUSAL 8ST"
        },
        {
            "start_char": 257,
            "end_char": 286,
            "label_": "AUTHOR",
            "text": "ABIGAIL VALERIA APAZA CONDORI"
        },
        {
            "start_char": 307,
            "end_char": 334,
            "label_": "ADVISOR",
            "text": "SERGIO EDDY VISCARRA ORTUÑO"
        },
        {
            "start_char": 356,
            "end_char": 360,
            "label_": "YEAR",
            "text": "2020"
        }
    ]
}
```


Fuente: Elaboración propia.

2.4.4 Estructuras de datos en spaCy

En la se muestra la jerarquía de estructuras de datos usadas en spaCy. Las mismas están documentadas por (ExplosionAI, s.f.-b), sin embargo tres de ellas que serán útiles para comprender el proceso de preparación de datos de entrenamiento explicado en la siguiente sección:

- `Token`: Contiene un token individual (por ejemplo una palabra, un signo de puntutación, un espacio en blanco, etc). Permite acceder a las anotaciones lingüísticas del mismo, por ejemplo a sus etiquetas NER o POS[7].
- `Doc`: Es una secuencia de objetos tipo `Token` . Permite acceder a las anotaciones lingüísticas de los mismos. Por ejemplo a sus etiquetas NER.
- `DocBin`: Es una colección de objectos `Doc`. El mismo, puede ser persistido en disco como un archivo `.spacy`, el cual luego puede ser usado para entrenar un modelo.

2.5 Modelos de Lenguaje Grande

Los modelos de lenguaje grande (LLM) son modelos de IA que generalmente (pero no necesariamente) se derivan de la arquitectura Transformer y están diseñados para comprender y generar lenguaje humano, código y mucho más. Estos modelos se entrenan con grandes cantidades de datos de texto, lo que les permite capturar las complejidades y matices del lenguaje humano.

Los LLM pueden realizar una amplia gama de tareas relacionadas con el lenguaje, como:

- Generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a tus preguntas de manera informativa.
- Resumir texto, escribir diferentes tipos de contenido creativo y responder a tus preguntas de manera informativa.

¿Cómo funcionan los LLMs?

1. **Entrenamiento:**
    - Los LLMs se construyen utilizando técnicas de aprendizaje profundo, específicamente una arquitectura de red neuronal llamada “transformadores”.

[7] Part-of-speech, etiquetas que indican a qué parte de una oración el token representa. Ejemplos de etiquetas en spaCy son: `NNP` para pronombres propios, `VBG` para verbos. Fuente ExplosionAI (s.f.-c).

    - Estos transformadores procesan el texto de una manera que les permite capturar relaciones de largo rango entre palabras y comprender el contexto.
    - El modelo se alimenta con cantidades masivas de datos de texto y aprende a predecir la siguiente palabra o frase en una secuencia.

2. **Inferencia:**

    - Una vez entrenados, los LLMs pueden recibir nuevas entradas de texto y generar salidas que sean tanto coherentes como relevantes al contexto.
    - También se les puede pedir que realicen tareas específicas, como traducción o resumen.

Componentes clave de los LLMs:

- **Redes neuronales:** El marco computacional subyacente inspirado en el cerebro humano.
- **Modelos de transformadores:** Un tipo específico de red neuronal adecuado para el procesamiento del lenguaje natural.
- **Capas de Embeddings:** Representan las palabras como vectores numéricos para capturar su significado y relaciones.
- **Capas de atención:** Se centran en partes específicas del texto de entrada para mejorar la comprensión.

Ejemplos de LLMs:

- GPT-3 (Generative Pre-trained Transformer 3) de OpenAI
- BERT (Bidirectional Encoder Representations from Transformers) de Google
- T5 (Text-to-Text Transfer Transformer) de Google
- Mistral-7B-Instruct-v0.1 de Mistral
- Llama 2 13B de Meta
- Gemini de Google

Aplicaciones de los LLMs:

- Traducción automática
- Resumen de texto
- Contestador de preguntas
- Chatbots
- Generación creativa de texto (poesía, código, guiones, etc.)

- Completado de código
- Redacción de correos electrónicos y cartas
- Otros

2.5.1 Clasificación de LLMs

Categorizar LLMs can be dificil, ya que no hay un única sistema universalmente aceptado. Sin embargo, hay varias formas de clasificarlos que pueden ayudar a comprender mejor sus diferencias y aplicaciones:

1. Por sus datos de entrenamiento:

- **De propósito general:** Entrenados en grandes conjuntos de datos de texto de dominio general, como libros, artículos de noticias y sitios web.
- **De dominio específico:** Entrenados en conjuntos de datos de texto de dominio específico, como documentos legales, registros médicos o publicaciones de redes sociales.
- **Ajustados:** Entrenados adicionalmente en conjuntos de datos específicos de tareas, como preguntas y respuestas o resúmenes de texto, para mejorar el rendimiento en esas tareas.

2. Por sus capacidades:

- **Generativos:** Capaces de generar texto, incluida la escritura creativa, el código y diferentes estilos de escritura.
- **Analíticos:** Centrados en comprender y analizar el texto, como la respuesta a preguntas, el análisis de sentimientos y la extracción de información.
- **Interactivos:** Diseñados para la conversación y el diálogo, utilizados en chatbots y asistentes virtuales.

3. Por el objetivo del preentrenamiento:

- **Autorregresivo (AR):** Estos modelos predicen el siguiente token en una secuencia basándose en los anteriores, centrándose principalmente en la generación de texto. Ejemplos incluyen GPT-3 y Jurassic-1 Jumbo.

- **Autoencoder (AE):** Estos modelos comprimen y luego reconstruyen los datos de entrada, con el objetivo de extraer y codificar representaciones latentes de los datos. Ejemplos incluyen BERT y RoBERTa.

- **Secuencia a secuencia (Seq2Seq):** Esta categoría combina elementos de AR y AE, usando a menudo un codificador (AE) para procesar la entrada y un decodificador (AR) para generar la salida. Ejemplos incluyen T5 y BART.

  Relación con las capacidades de los LLMs:

  - **Los modelos AR sobresalen en tareas generativas** como escribir en diferentes estilos, completar código y generar contenido creativo.
  - **Los modelos AE son más competentes en tareas analíticas** como la clasificación de texto, la recuperación de información y la respuesta a preguntas.
  - **Los modelos Seq2Seq ofrecen flexibilidad para la generación y el análisis**, dependiendo de la arquitectura específica y los objetivos del entrenamiento.

2.5.2 Ingeniería de prompts

En el dominio de los modelos de lenguaje grandes (LLMs), la **ingeniería de prompts** es el arte de crear instrucciones (prompts) para guiar la salida de los modelos. Al construir cuidadosamente estos prompts, podemos decirle al modelo qué hacer, cómo hacerlo y qué esperar, lo que conduce a resultados más precisos, relevantes e incluso creativos.

Dentro de la ingeniería de prompts, existen tres estrategias principales para guiar la salida de un LLM:

- **Aprendizaje sin ejemplos (Zero-shot):** En este enfoque, un Modelo de Lenguaje Grande (LLM) realiza una tarea sin ningún prompt o instrucción explícita. Esto significa que el modelo debe aprovechar su conocimiento interno y su comprensión del mundo para completar la tarea basándose únicamente en los datos de entrada. Por ejemplo, podría pedirle al modelo que "Genere una descripción de producto para este nuevo teléfono inteligente", sin proporcionar ningún ejemplo de descripción de producto. Este enfoque es rápido y versátil, pero puede producir resultados imprecisos o irrelevantes.

- **Aprendizaje con un ejemplo (One-shot):** En este enfoque, un Modelo de Lenguaje Grande (LLM) realiza una tarea basándose en un solo ejemplo de entrada-salida proporcionado dentro del prompt. Este único ejemplo actúa como una plantilla o guía para que el modelo generalice y aplique su conocimiento a nuevas entradas no vistas.

In one-prompt learning (OPL), a large language model (LLM) learns to perform a task based on a single

exemplary input-output pair provided within the prompt. This single example acts as a template or guide for the model to generalize and apply its knowledge to new, unseen inputs.

- **Aprendizaje con pocos ejemplos (Few-shot):** Este enfoque es similar al aprendizaje con un ejemplo, pero proporciona varios ejemplos de entrada-salida dentro del prompt. Esto permite que el modelo aprenda de múltiples ejemplos y generalice aún más su conocimiento a nuevas entradas no vistas.

La elección de la estrategia de ingeniería de prompts adecuada depende de la tarea en cuestión y los recursos disponibles. El aprendizaje sin ejemplos ofrece versatilidad pero puede carecer de precisión, mientras que el aprendizaje con un ejemplo y con pocos ejemplos, aunque requiere ejemplos adicionales, puede conducir a resultados más precisos y personalizados.

2.5.3 Ajuste fino

El **ajuste fino del aprendizaje** es una técnica que permite a un modelo de lenguaje grande (LLM) transferir el conocimiento adquirido durante su pre-entrenamiento en un conjunto de datos masivo a una nueva tarea específica. Esto se logra:

- **Congelando la mayoría de las capas pre-entrenadas:** La gran mayoría de las capas pre-entrenadas del LLM (por ejemplo, las capas inferiores) se mantienen sin cambios, conservando el conocimiento general que ya poseen.
- **Re-entrenando las capas finales:** Solo la última capa (o un pequeño subconjunto de la red) se re-entrena en el nuevo conjunto de datos más pequeño específico para la tarea deseada. Este ajuste fino permite al LLM adaptar su conocimiento preexistente a los matices específicos de la nueva tarea.

Desde una perspectiva técnica, el ajuste fino del aprendizaje aprovecha:

- **Aprendizaje por transferencia:** El conocimiento general adquirido en la fase de pre-entrenamiento se transfiere a la nueva tarea, reduciendo la necesidad de grandes cantidades de datos de entrenamiento.
- **Adaptación gradual:** Las capas finales reciben información nueva y refinan su comprensión en base a los datos específicos de la tarea, construyendo sobre la base del conocimiento de las capas pre-entrenadas.

- **Optimización de parámetros:** Los algoritmos de aprendizaje ajustan los parámetros de las capas re-entrenadas para optimizar el rendimiento en la nueva tarea.

El ajuste fino presenta numerosas ventajas:

- **Eficiencia de datos:** Requiere significativamente menos datos en comparación con el entrenamiento de un nuevo LLM desde cero, haciéndolo ideal para escenarios con datos de entrenamiento limitados.
- **Entrenamiento más rápido:** Aprovechar el conocimiento pre-entrenado conduce a una convergencia más rápida y un rendimiento mejorado en comparación con el entrenamiento desde cero.
- **Adaptabilidad a la tarea:** Se puede aplicar a una amplia gama de tareas personalizando el proceso de ajuste fino y los datos.

Sin embargo, el ajuste fino también enfrenta desafíos:

- **Sobreajuste a la tarea:** Enfatizar demasiado la tarea específica durante el ajuste fino puede conducir a una disminución del rendimiento en otras tareas o aplicaciones más generales.
- **Olvido catastrófico:** El conocimiento pre-entrenado podría ser parcialmente sobrescrito durante el ajuste fino, lo que podría afectar el rendimiento en las tareas aprendidas previamente.
- **Ajuste de hiperparámetros:** Elegir los parámetros óptimos para el ajuste fino puede ser desafiante y requerir experimentación.

A pesar de estos desafíos, el ajuste fino sigue siendo una técnica poderosa y versátil para el aprendizaje de LLMs, especialmente cuando los datos son limitados o las tareas están bien definidas. Su capacidad para aprovechar el conocimiento pre-entrenado y adaptarse rápidamente a nuevas tareas lo convierte en una herramienta valiosa para diversas aplicaciones de procesamiento del lenguaje natural.

2.5.3.1 PEFT

El **Ajuste Fino con Eficiencia de Parámetros (PEFT)** es una técnica para adaptar los Modelos de Lenguaje Grande (LLMs) a tareas específicas con un costo computacional y una huella de memoria mucho menores que el ajuste fino tradicional. Logra esto de la siguiente manera:

1. **Actualización selectiva de parámetros:** A diferencia del ajuste fino estándar donde **todos** los

parámetros del modelo se vuelven a entrenar, PEFT se enfoca en actualizar solo un **pequeño subconjunto** de parámetros considerados más relevantes para la nueva tarea. Este enfoque selectivo reduce drásticamente la cantidad de cálculos y memoria necesarios.

2. Aprovechamiento de modificadores de bajo rango: En lugar de modificar directamente las matrices de peso completo, PEFT utiliza modificadores de bajo rango, esencialmente matrices más pequeñas que se agregan o multiplican a los pesos pre-entrenados. Estos modificadores inyectan conocimiento específico de una tarea minimizando los cambios en las capacidades subyacentes del LLM.

3. Diferentes técnicas: Existen varias técnicas PEFT, cada una con sus propias fortalezas y debilidades, como:

- **LoRA (Adaptación de bajo rango):** Utiliza modificadores de bajo rango para adaptar capas individuales.
- **Ajuste de prefijos:** Se centra en optimizar un pequeño prefijo agregado a la secuencia de entrada, guiando la salida del modelo.
- **Módulos adaptadores:** Introduce pequeñas redes "adaptadoras" encima de capas pre-entrenadas específicas, inyectando conocimiento específico de la tarea sin modificar la funcionalidad central del LLM.

Beneficios de PEFT:

- **Costo computacional reducido:** El entrenamiento es significativamente más rápido y requiere menos hardware en comparación con el ajuste fino completo.
- **Eficiencia de memoria:** Requiere menos memoria para almacenar el modelo actualizado, permitiendo la implementación en dispositivos más pequeños.
- **Potencial de generalización:** Las actualizaciones selectivas potencialmente ayudan a mantener el conocimiento general del LLM y mejoran la capacidad de adaptarse a nuevas tareas.

Desafíos de PEFT:

- **Idoneidad de la tarea:** PEFT podría no ser igualmente efectivo para todas las tareas, particularmente aquellas que requieren cambios significativos en las representaciones internas del LLM.
- **Interpretabilidad:** Entender cómo PEFT modifica el comportamiento del LLM puede ser

complejo en comparación con el ajuste fino completo.

2.5.4 Evaluación de LLMs

Evaluar el rendimiento de los LLMs en tareas de NLP puede ser un proceso complejo y multifacético. No existe una métrica única "perfecta", y la elección de los métodos de evaluación depende de la tarea específica y las cualidades deseadas. Aquí hay una descripción general de algunos enfoques comunes:

Métricas automáticas:

- **Perplexity:** Mide qué tan bien el modelo predice la siguiente palabra en una secuencia. Una perplejidad más baja sugiere una mayor precisión de predicción.
- **Accuracy:** Calcula el porcentaje de predicciones que coinciden con la respuesta de referencia, a menudo utilizado para tareas de clasificación.
- **Precision y Recall:** Evalúa la capacidad del modelo para identificar correctamente verdaderos positivos y verdaderos negativos, particularmente relevante para la respuesta a preguntas o la recuperación de información.
- **Rouge Score:** Compara el texto generado con el texto de referencia, evaluando aspectos como la superposición de oraciones y la similitud semántica.
- **BLEU Score:** Similar a Rouge pero se centra en la superposición de n-gramas, destacando la fluidez y la corrección gramatical.

2.5.4.1 ROUGE

ROUGE, que significa Recall-Oriented Understudy for Gisting Evaluation (Estudio de referencia orientado a la recuperación para la evaluación de la esencia), es un conjunto de métricas utilizadas para evaluar la calidad de los resúmenes de texto generados por modelos de NLP. Es particularmente popular para tareas como traducción automática, resumen de texto y generación de respuestas a diálogos.

ROUGE se enfoca en medir la **superposición** entre un resumen generado y un resumen de referencia, generalmente escrito por humanos. Lo hace comparando n-gramas, que son secuencias de palabras consecutivas. Las diferentes variantes de ROUGE utilizan diferentes n-gramas para evaluar la superposición en distintos niveles:

- **ROUGE-1:** Utiliza unigramas (palabras individuales)
- **ROUGE-2:** Utiliza bigramas (secuencias de dos palabras)
- **ROUGE-L:** Utiliza la subsecuencia común más larga de palabras, independientemente del orden.
- **ROUGE-N:** Utiliza una combinación de n-gramas para obtener una puntuación más completa.

Luego, ROUGE calcula la **Precisión, el Recall y el F1-score** para cada nivel de n-gramas.

La **Precisión** mide la **proporción de n-gramas en el resumen generado que también aparecen en el resumen de referencia**. En términos más sencillos, dice qué parte del resumen generado es relevante para el texto original. Se calcula como:

$$ROUGE_{Precision} = \frac{\text{Número de n-gramas en común}}{\text{Número de n-gramas en el resumen generado}}$$

El **Recall** mide la **proporción de n-gramas en el resumen de referencia que también aparecen en el resumen generado**. Dice qué tanta información clave del texto original se captura en el resumen generado. Se calcula como:

$$ROUGE_{Recall} = \frac{\text{Número de n-gramas en común}}{\text{Número de n-gramas en el resumen de referencia}}$$

El **F1-score** es una media armónica de la precisión y la recuperación, proporcionando una medida equilibrada de qué tan bien el modelo captura la relevancia y la integridad en su resumen. Se calcula como:

$$ROUGE_{F1} = 2 * \frac{ROUGE_{Precision} * ROUGE_{Recall}}{ROUGE_{Precision} + ROUGE_{Recall}}$$

2.5.5 Benchmarks

Los Benchmarks son herramientas cruciales para evaluar el rendimiento de los modelos de lenguaje grande (LLM) en tareas específicas de procesamiento de lenguaje natural (PLN). Actúan como **pruebas estandarizadas** que comparan diferentes modelos en condiciones controladas, permitiendo comparaciones objetivas y perspectivas sobre sus fortalezas y debilidades.

Existen muchos Benchmarks de LLM diferentes, cada uno centrado en un aspecto particular de la PLN, como:

- **Traducción automática:**
  - **MT-Bench:** Este Benchmark evalúa la calidad de la traducción automática en varios factores como la fluidez, la precisión y la correspondencia de estilo. Incluye conjuntos de pruebas diversos que abarcan diferentes idiomas y dominios.
  - **WMT:** El Taller sobre Traducción Automática es un evento anual con tareas compartidas que sirven como Benchmarks para la investigación en traducción automática. Se centra en desafíos específicos de traducción, como idiomas con pocos recursos y adaptación de dominio.
- **Respuesta a preguntas:**
  - **SQuAD:** El conjunto de datos de preguntas y respuestas de Stanford es un Benchmark popular para tareas de comprensión de lectura. Implica responder preguntas basadas en pasajes de texto proporcionados.
  - **TriviaQA:** Este Benchmark evalúa la respuesta a preguntas de hecho, con preguntas que requieren un amplio conocimiento y razonamiento sobre un corpus de texto extenso.
- **Diálogo y resumen:**
  - **Bliu:** Este Benchmark se centra en la generación de lenguaje basado en diálogos, probando la capacidad del modelo para responder a las indicaciones de manera natural e informativa.
  - **CNN/Daily Mail Summarization:** Estos conjuntos de datos evalúan la capacidad de los LLM de generar resúmenes concisos e informativos de artículos de noticias.

2.5.5.0.1 MT-Bench

MT-Bench evalúa la calidad de la traducción automática en múltiples dimensiones, más allá de la simple precisión palabra por palabra. Incluye métricas como gramaticalidad, fluidez y fidelidad estilística, proporcionando una imagen más matizada del rendimiento de la traducción.

2.5.5.0.2 MMLU

El conjunto de datos de entrenamiento no supervisado para múltiples tareas sin etiquetar presenta un desafío diferente. En lugar de centrarse en una sola tarea como la traducción, MMTU incluye un

conjunto diverso de datos de texto sin etiquetar de varios dominios. Los LLM entrenados en MMTU se evalúan según su capacidad para aprender representaciones de lenguaje de propósito general que se pueden aplicar a diversas tareas de PLN sin necesidad de ajustes adicionales.

2.6 Mistral-7B-Instruct-v0.1

Mistral-7B-Instruct-v0.1 es un modelo de lenguaje grande (LLM) diseñado específicamente para la interacción conversacional y la ejecución de tareas guiadas por instrucciones. Construido sobre la base del modelo de texto generativo `Mistral-7B-v0.1`, se ha optimizado aún más utilizando distintos conjuntos de datos de conversaciones disponibles públicamente. Este proceso de "fine-tuning" mejora su capacidad para:

- Comprender y responder a peticiones e instrucciones en lenguaje natural.
- Participar en diálogos informativos y atractivos.
- Seguir instrucciones específicas y completar tareas tal como se indiquen.

Aquí algunas características clave de Mistral-7B-Instruct-v0.1:

- **Arquitectura del modelo:** Utiliza un modelo de Transformador con funciones como *group-query attention* y *sliding-window attention*, lo que le permite procesar eficazmente el contexto conversacional y las dependencias de largo rango en el lenguaje.
- **"Fine-tuning" para el seguimiento de instrucciones:** El modelo se entrenó en conjuntos de datos que contenían conversaciones, diálogos y tareas basadas en instrucciones, mejorando su capacidad para entender y ejecutar instrucciones.
- **Limitaciones:** Si bien es prometedor, Mistral-7B-Instruct-v0.1 aún está en desarrollo y carece de mecanismos de moderación para salidas potencialmente dañinas o sesgadas.

En general, Mistral-7B-Instruct-v0.1 tiene potencial para diversas aplicaciones de IA conversacional, como:

- **Chatbots y asistentes virtuales:** Con su capacidad para comprender consultas en lenguaje natural y seguir instrucciones, podría impulsar interfaces conversacionales más dinámicas y útiles.
- **Narrativa interactiva y entornos de juego:** Su flexibilidad para responder a indicaciones e instrucciones podría facilitar experiencias interactivas atractivas e inmersivas.

- **Tecnología educativa:** Guiando a los usuarios a través de tareas y respondiendo preguntas, podría convertirse en una herramienta valiosa para el aprendizaje personalizado.

2.7 Herramientas de software para el desarrollo de proyectos de Machine Learning

2.7.1 Python

Python es un lenguaje de programación de alto nivel y uso general conocido por su facilidad de lectura, versatilidad y amplias bibliotecas para ciencia de datos, aprendizaje automático (ML) y procesamiento de lenguaje natural (NLP). Ofrece varias ventajas clave que lo convierten en una opción ideal para construir proyectos de ML y NLP:

1. Facilidad de uso y legibilidad:

- Sintaxis clara: La sintaxis de Python se asemeja al lenguaje natural, lo que la hace fácil de aprender, escribir y entender, incluso para quienes recién comienzan a programar.
- Tipado dinámico: Infere automáticamente los tipos de datos, reduciendo la verbosidad del código y permitiendo un desarrollo rápido.
- Entorno interactivo: Facilita la experimentación y prueba de fragmentos de código, promoviendo un flujo de trabajo rápido.

2. Rico ecosistema de bibliotecas y frameworks:

- NumPy y Pandas: Proporcionan herramientas poderosas para cálculos numéricos y manipulación de datos, formando la base del análisis de datos y las tareas de aprendizaje automático.
- TensorFlow y PyTorch: Permiten el desarrollo de modelos de aprendizaje profundo, que están a la vanguardia de los avances en NLP.
- NLTK y spaCy: Se especializan en tareas de procesamiento de lenguaje natural, proporcionando herramientas para tokenización, análisis sintáctico, reconocimiento de entidades nombradas, análisis de sentimiento y más.

3. Fuerte comunidad y apoyo:

- Comunidad activa: Python cuenta con una comunidad grande y activa de desarrolladores,

investigadores y entusiastas, fomentando el intercambio de conocimientos y la colaboración.

- Amplios recursos: Abundancia de tutoriales en línea, documentación y foros que brindan apoyo y orientación.

4. Integración perfecta con otras herramientas:

- Bibliotecas de visualización: Seaborn y Matplotlib permiten la creación de visualizaciones de datos perspicaces, lo que ayuda a comprender el modelo y comunicar los resultados.
- Jupyter Notebooks: Permite la ejecución de comandos Python de manera interactiva en forma de un "cuaderno", lo que facilita el desarrollo de códigos complejos.

2.7.2 Jupyter Notebooks

Un Jupyter Notebook es una aplicación web interactiva para crear y compartir "documentos computacionales". Combina código, resultados, visualizaciones y texto narrativo en un solo documento, convirtiéndola en una herramienta popular para la exploración de datos y el aprendizaje automático (Jupyter, s.f.).

A continuación se mencionan algunas de las características, beneficios y aplicaciones de los Jupyter Notebooks.

Características principales:

- **Interactividad:** Se puede ejecutar celdas de código y ver los resultados inmediatamente, lo que permite una exploración y experimentación iterativa.
- **Contenido enriquecido:** Incluye varios elementos como código, texto con formato, imágenes, gráficos y formatos multimedia dentro del mismo documento.
- **Múltiples lenguajes:** Admite una amplia gama de lenguajes de programación como Python, R, Julia y más, ampliando su versatilidad.
- **Código abierto:** Es de libre acceso y está en constante evolución, con una gran comunidad activa que contribuye a su desarrollo.
- **Documentación:** Facilita la fácil compartición y colaboración al combinar código y explicaciones en un solo documento.

Beneficios:

- **Prototipado rápido:** Prueba y refina rápidamente ideas a través de la exploración visual

interactiva.

- **Reproducibilidad:** Comparte notebooks con otros para garantizar la transparencia y replicabilidad de tu trabajo.
- **Educación y enseñanza:** Ideal para aprender nuevos conceptos y demostrar algoritmos paso a paso.
- **Presentaciones e informes:** Combina código, análisis y resultados en un solo documento para una comunicación efectiva.

Aplicaciones:

- **Ciencia de datos y aprendizaje automático:** Usado para la exploración de conjuntos de datos, desarrollo de modelos y visualización de resultados.
- **Computación científica:** usado para realizar cálculos numéricos, analizar datos y presentar hallazgos.
- **Desarrollo de software:** Usado en prototipado, pruebas y documentación de código.
- **Educación y formación:** Ideal para aprender nuevos conceptos y demostrar algoritmos paso a paso.
- **Presentaciones e informes:** Combina código, análisis y resultados en un solo documento para una comunicación efectiva.

2.7.3 DataSpell

DataSpell es un entorno de desarrollo integrado (IDE) diseñado específicamente para Ciencia de Datos y Aprendizaje Automático, desarrollado por JetBrains. Se basa en las capacidades del popular IDE PyCharm de JetBrains y las adapta a las necesidades de los científicos de datos y profesionales del Aprendizaje Automático (DataSpell, s.f.).

Sus características clave son:

- **Integración con Jupyter Notebook:** Se puede abrir, edita y ejecuta sin problemas Jupyter Notebooks dentro del IDE, combinando la exploración interactiva de Notebooks con las sólidas funciones de desarrollo de un IDE.
- **Exploración de datos interactiva:** Se puede visualizar conjuntos de datos y matrices, explorar datos de forma interactiva con filtrado y clasificación, y crear visualizaciones con integraciones para bibliotecas como Matplotlib y Plotly.

- **Asistente de sugerenncias de código inteligente:** Ofrece finalización de código o “code completion” en contexto, navegación, refactorización y depuración para Python.
- **Integración con control de versiones:** Trabaja con Git para tomar registros cambios, colaborar con otros y administrar el historial del proyecto de manera eficiente.
- **Entornos personalizables:** Facilita la administración de múltiples entornos virtuales de Python y dependencias del proyecto para la reproducibilidad.

Las aplicaciones de DataSpell incluyen:

- Exploración y análisis de datos: Visualización y comprensión de conjuntos de datos.
- Modelado de aprendizaje automático: Construir, entrenar y evaluar modelos de aprendizaje automático.
- Seguimiento de experimentos: Administrar y comparar diferentes configuraciones de modelos y resultados.
- Ingeniería de datos: Limpiar, transformar y preparar datos para el análisis.
- Informes y paneles interactivos: Creación de visualizaciones dinámicas y compartir conocimientos.

2.7.4 OpenCV

OpenCV, es una biblioteca ampliamente utilizada para la visión por computadora (OpenCV, s.f.). Ofrece un conjunto completo de funciones y algoritmos para tareas como:

- Procesamiento de imágenes
- Procesamiento de video
- Detección de objetos
- Reconocimiento facial
- Seguimiento de movimiento

Las ventajas de usar OpenCV incluyen:

- **Gratuito y de código abierto:** Disponible para uso comercial y no comercial bajo una licencia BSD.
- **Multiplataforma:** Funciona en Windows, Linux, macOS, Android e iOS.
- **Bien documentado y con soporte:** Amplia documentación, tutoriales y una gran comunidad de

soporte.

- **Alto rendimiento:** Optimizado para aplicaciones en tiempo real.
- **Amplia gama de algoritmos:** Cubre un amplio espectro de tareas de visión por computadora.

2.7.5 Tesseract

Tesseract es un motor de reconocimiento óptico de caracteres (OCR) de código abierto que extrae texto de imágenes. Fue desarrollado originalmente por Hewlett-Packard en los años 80 y desde 2005 ha sido patrocinado por Google. Es conocido por su precisión, versatilidad y soporte para más de 100 idiomas (Tesseract, s.f.).

A continuación se mencionan algunas de las características, ventajas y aplicaciones de Tesseract.

- **Reconocimiento de texto preciso:** Tesseract tiene la reputación de ser uno de los motores OCR de código abierto más precisos, especialmente para texto bien formateado y documentos impresos.
- **Soporte multilingüe:** Admite más de 100 idiomas, lo que lo hace adecuado para una amplia gama de tareas de extracción de texto en diferentes idiomas.
- **Formatos de imagen:** Puede manejar varios formatos de imagen, incluidos PNG, JPEG, TIFF y PDF.
- **Formatos de salida:** Puede generar texto de salida en formatos de texto plano, HOCR (HTML), PDF, TSV y ALTO, lo que permite la integración con diferentes aplicaciones.
- **Línea de comandos y API:** Se puede usar como una herramienta de línea de comandos para procesamiento por lotes y como una biblioteca integrada en otras aplicaciones de software.
- **Integración con lenguajes de programación:** Tesseract tiene clientes para varios lenguajes de programación, incluidos C++, Python, Java y C#, lo que lo hace accesible a desarrolladores con diferentes necesidades.

Aquí hay algunas aplicaciones comunes de Tesseract:

- **Digitalización de documentos:** Convertir documentos escaneados, PDF e imágenes en texto editable para archivar, buscar y hacerlos accesibles.
- **Extracción de texto de imágenes:** Extraer texto de fotos, capturas de pantalla y otras imágenes para diversos fines, como extracción de datos, traducción y análisis.
- **Procesamiento de datos basado en texto:** Automatizar tareas que involucran la extracción de

texto de imágenes, como la entrada de datos, la recuperación de información y el análisis de texto.

2.7.6 Pandas

Pandas es una popular biblioteca de Python para el análisis y la manipulación de datos. Brinda una forma flexible, eficiente y de alto nivel para trabajar con datos tabulares, Dataframes y Series (NVIDIA, s.f.).

A continuación se mencionan algunas de las características, beneficios y aplicaciones de Pandas.

- **Estructuras de datos:** Proporciona estructuras de datos esenciales como DataFrames (tablas 2D) y Series (arreglos 1D) para una organización y manipulación de datos eficiente.
- **Carga y limpieza de datos:** Facilita la carga de datos de diversas fuentes como CSV, Excel, bases de datos SQL y más. Ofrece herramientas para la limpieza y el preprocesamiento de datos, incluido el manejo de valores faltantes, valores atípicos y tipos de datos.
- **Análisis de datos:** Brinda numerosas funciones estadísticas y matemáticas para analizar datos, incluyendo estadísticas descriptivas, correlaciones, pruebas de hipótesis y agregación de datos.
- **Visualización de datos:** Ofrece amplias capacidades de visualización de datos con funciones integradas para crear diagramas, gráficos e histogramas para explorar y comprender las relaciones entre los datos.
- **Combinación y unión de datos:** Combina y une fácilmente datos de diferentes fuentes en función de llaves comunes u otras condiciones.
- **Alto rendimiento:** Optimizado para manejar grandes conjuntos de datos de manera eficiente con operaciones vectorizadas y estructuras de datos eficientes en memoria.

Las aplicaciones de Pandas incluyen:

- **Finanzas:** Analizar datos financieros, construir modelos de inversión y evaluar riesgos.
- **Ciencia e ingeniería:** Analizar datos científicos, realizar pruebas estadísticas y visualizar resultados.
- **Ciencias sociales:** Analizar datos de encuestas, realizar estudios demográficos y construir modelos estadísticos.
- **Marketing y negocios:** Analizar datos de clientes, construir campañas de marketing y medir el rendimiento.

- **Ciencias de la salud:** Analizar registros médicos, identificar tendencias de pacientes y predecir resultados.

2.7.7 Huggingface

Hugging Face es una plataforma multifacética especializada en inteligencia artificial, con un enfoque en:

- **Intercambio y alojamiento de modelos:** Brinda un repositorio central para subir, descargar y explorar modelos de IA de código abierto en múltiples tareas como generación de texto, clasificación de imágenes y respuesta a preguntas. Esto permite a investigadores y desarrolladores construir sobre el trabajo existente y acelerar sus proyectos de IA.
- **Biblioteca Transformers:** Ofrece la popular biblioteca Transformers, un potente kit de herramientas para construir y entrenar modelos de vanguardia de procesamiento de lenguaje natural (NLP). Agiliza el desarrollo al proporcionar modelos pre-entrenados, scripts de entrenamiento y utilidades para ajustar modelos en tareas específicas.
- **Plataforma impulsada por la comunidad:** Fomenta una próspera comunidad de entusiastas, investigadores y desarrolladores de IA a través de foros, debates y competiciones. Este entorno colaborativo promueve el intercambio de conocimientos, la mejora de modelos y nuevas aplicaciones de IA.

Aquí hay algunas características y beneficios clave de Hugging Face:

- **Gran variedad de modelos:** Más de 350,000 modelos pre-entrenados cubriendo diversas tareas como resumen de texto, análisis de sentimiento y generación de código.
- **Acceso y uso fácil:** Interfaz intuitiva para explorar, descargar y usar modelos con mínima experiencia técnica requerida.
- **Hub de modelos:** Aloja modelos aportados por los usuarios con detalles sobre su rendimiento, datos de entrenamiento y posibles aplicaciones.
- **Espacios:** Permite experimentar con modelos dentro de entornos interactivos, permitiendo que otros los prueben y proporcionen feedback.
- **Tabla de clasificación:** Sigue el rendimiento de diferentes modelos en tareas específicas, fomentando una competencia saludable e impulsando la innovación.
- **Código abierto y colaborativo:** Incentiva las contribuciones de código abierto y fomenta un

espíritu colaborativo dentro de la comunidad de IA.

La documentación oficial de Hugging Face describe en detalle las características y funcionalidades de la plataforma[8].

2.7.8 MLFlow

MLflow es una plataforma de código abierto creada para gestionar todo el ciclo de vida del aprendizaje automático. Proporciona un conjunto de herramientas y API que se pueden usar para:

- **Realizar un seguimiento de experimentos:** registra y consulta los metadatos de los experimentos de aprendizaje automático, como los parámetros utilizados, las métricas registradas y los artefactos producidos.
- **Administrar modelos:** empaqueta modelos de aprendizaje automático en un formato estándar para facilitar su intercambio y despliegue en producción.
- **Desarrollar proyectos reproducibles:** empaqueta el código, datos y entorno de aprendizaje automático para garantizar la consistencia y facilidad de compartir.

Los beneficios de usar MLflow incluyen:

- **Mayor reproducibilidad:** facilita replicar experimentos, crucial para depurar problemas y asegurar la fiabilidad de tus modelos.
- **Gestión simplificada de modelos:** ofrece un lugar centralizado para almacenar, organizar y desplegar los modelos.

[8]Aunque OpenAI no ha revelado el tamaño real de su modelo GPT3.5, se rumora que tiene el mismo tamaño que su versión 3, osea 175 mil millones de parámetros.

3 Marco práctico

En concordancia con el marco de trabajo "*Metodología Fundamental para la Ciencia de Datos*" de IBM (ver Sección 2.1) el proyecto se realizó en las siguientes etapas:

- **Etapa 1:** Comprensión del negocio
- **Etapa 2:** Enfoque analítico
- **Etapa 3:** Requisitos de datos
- **Etapa 4:** Recopilación de datos
- **Etapa 5:** Comprensión de datos
- **Etapa 6:** Preparación de datos
- **Etapa 7:** Modelado
- **Etapa 8:** Evaluación

Existen 2 pasos adicionales en el marco que no se llevaron a cabo, "Implementación" y "Monitoreo", debido a que la puesta a producción de los modelos desarrollados estaba fuera del alcance del presente proyecto de grado.

3.1 Etapa 1: Comprensión del negocio

Para la comprensión del negocio se realizaron entrevistas con los stakeholders del proyecto. Las entrevistas fueron realizadas en forma de reuniones formales e informales, en las que se conversó sobre el problema a resolver, los requerimientos del sistema a desarrollar, y las restricciones del proyecto.

Los stakeholders del proyecto fueron:

1. **Personal administrativo de la biblioteca de la Facultad de Ingeniería:** La razón por la que se constituyen en parte interesada del proyecto es porque son ellos quien se beneficiarían directamente de la automatización del registro de documentos académicos, una de las morosas actividades que realizan a diario.

2. **Personal administrativo del Departamento de Tecnologías de Información y Comunicación (DTIC) de la UMSA:** El DTIC implementaría el sistema final en base a los modelos ML desarrollados en el presente proyecto y lo desplegaría a producción, si así deciden hacerlo.

¿Cual es el problema a resolver?

El problema a resolver es la automatización del registro de documentos académicos en la biblioteca de la Facultad de Ingeniería de la Universidad Mayor de San Andrés (UMSA). Actualmente el registro de documentos académicos es un proceso manual y moroso, que requiere de la intervención de personal administrativo de la biblioteca. El proceso consiste en extraer los datos bibliográficos de la carátula de los documentos y registrarlos en una base de datos. Los datos bibliográficos extraídos son:

- Facultad
- Carrera
- Título del proyecto
- Nombre del o de los autores
- Nombre del o de los tutores
- Año de publicación

Adicionalmente, se requiere extraer las palabras clave del resumen del documento. Las palabras clave son una lista de palabras o frases que mejor describen los tópicos del documento.

¿Como se puede hacer el proceso mas eficiente?

El proceso puede ser automatizado mediante el uso de modelos de Machine Learning (ML) y Procesamiento de Lenguaje Natural (NLP). Los modelos ML pueden ser entrenados para extraer los datos bibliográficos y las palabras clave de los documentos académicos.

¿Cuál es la meta del negocio?

La meta del negocio es la automatización del registro de documentos académicos en la biblioteca de la Facultad de Ingeniería de la UMSA.

3.1.1 Requerimientos

Junto a los Stakeholders se determinaron los siguientes requerimientos para el sistema a desarrollar:

3.1.1.1 Requerimientos funcionales

1. **Extracción de datos desde la carátula:** El sistema debe ser capaz de extraer desde la carátula de los documentos académicos los siguientes datos bibliográficos:

1. Facultad
2. Carrera
3. Título del proyecto
4. Nombre del o de los autores
5. Nombre del o de los tutores
6. Año de publicación

2. **Extracción de palabras clave (también llamados descriptores):** El sistema debe ser capaz de extraer desde el texto del resumen del documento una lista de palabras o frases claves que mejor describan los tópicos del proyecto.
3. **Tipos de documentos:** El sistema debe ser capaz de extraer los datos desde documentos de fin de grado, como ser: Proyectos de Grado, Tesis de Grado, Tesis de Maestría y Memorias Laborales.
4. **Precisión:** Se espera una alta precisión del sistema. Aunque no se espera una precisión del 100%, se espera que los datos extraídos requieran mínimas correcciones.

3.1.1.2 Requerimientos no funcionales

1. **Rendimiento:** El sistema debe extraer los datos desde un documento en menos de 1 minuto.
2. **Seguridad:** Por políticas de la Universidad, el sistema debe poder ser alojado en un servidor de la institución.
3. **Usabilidad:** El sistema resultante debe poder ser desplegado en un servidor convencional, no requiriendo la adquisición de hardware especializado costoso.

3.1.1.3 Restricciones

1. **Presupuesto:** El sistema no debe representar un costo recurrente adicional, por lo tanto no debe estar basado en servicios de suscripción en la nube.

3.2 Etapa 2: Enfoque analítico

Como puede el ML y NLP ayudar a resolver el problema?

El ML y NLP pueden ayudar a resolver el problema mediante la construcción de un sistema que extraiga los datos bibliográficos y las palabras clave de los documentos académicos. El sistema puede ser construido en base a modelos de ML y NLP.

¿Que tipo de modelos ML y NLP pueden ser usados?

La **extracción de datos desde la carátula de los documentos** se constituye en un problema de Extracción de Información, tarea ampliamente estudiada por el NLP. En específico esta tarea puede ser abordada por el Reconocimiento de Entidades Nombradas (NER por sus siglas en inglés), tal como se vió en en unos de los trabajos descritos en la Sección 1.1.1. Por tanto para automatizar la extracción de estos datos se optó por desarrollar un sistema NER basado en modelos de ML.

La **extracción de palabras clave**, es también un problema de Extracción de Información, sin embargo a diferencia del caso anterior, las palabras clave no necesariamente son un extracto de porciones de texto del resumen, si no más bien es una lista de palabras o frases que mejor describen los tópicos del documento. Bajo ese entendido, este problema se vió como un problema de *síntesis* o *resumen de información*. Dado que en los últimos años los Modelos de Lenguaje Grande (LLM por sus siglas en inglés) se han destacado por sus habilidades de sintetizar o resumir textos con niveles de calidad nunca antes vistos, para la extracción de palabras clave se optó por desarrollar un sistema de generación de palabras clave basado en un LLM pre-entrenado en resumir texto.

¿Como la salida de los modelos responderá a las necesidades del negocio?

La salida de los modelos ML responderá a las necesidades del negocio en la medida en que los datos extraídos sean correctos. Por tanto, la precisión de los modelos es un factor clave para que la salida de los modelos responda a las necesidades del negocio.

¿Como se evaluará la precisión de los modelos?

La precisión de los modelos será evaluada mediante el uso de métricas de evaluación de modelos de ML. Para el caso del modelo de extracción de metadatos, se usará las métricas Precision, Recall y F1-Score. Para el caso del modelo de generación de palabras clave, se usará la métrica ROUGE.

¿Cuales son los niveles de precisión que cumplen con los requerimientos del negocio?

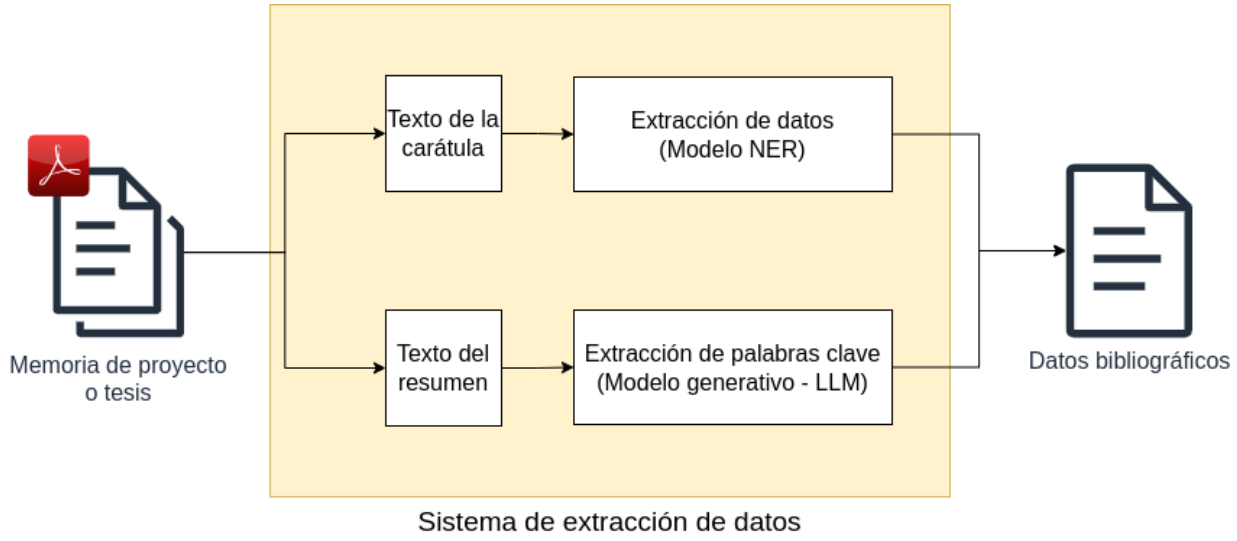
Los niveles de precisión que cumplen con los requerimientos del negocio son:

- **Extracción de metadatos:** Precision > 90%, Recall > 90%, F1-Score > 90%
- **Generación de palabras clave:** ROUGE ~= 0.5

3.2.1 Descripcion de los subsistemas a desarrollar

La Figura 16 muestra el diagrama de bloques a alto nivel donde se indican los dos subsistemas desarrollados indicando sus entradas y salidas.

Figura 16 - Diseño general del sistema a desarrollar.

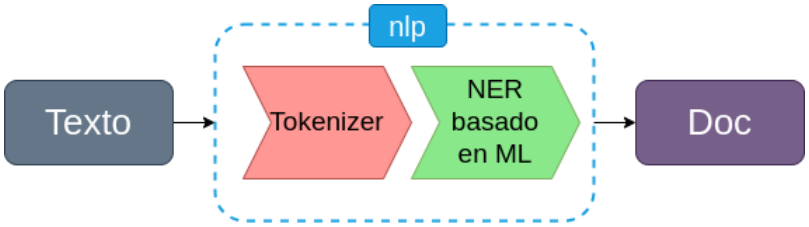


Fuente: Elaboración propia.

3.2.1.1 Subsistema de extracción de metadatos

Este subsistema consistirá en un pipeline de SpaCy compuesto de 2 componentes: un componente "tokenizer" y un componente "ner" (ver Figura 17).

Figura 17 - Pipeline de SpaCy para la extracción de metadatos.



Fuente: Elaboración propia.

La entrada del pipeline será un texto de la carátula de un documento académico, y la salida será un el mismo texto pero con las entidades reconocidas marcadas con etiquetas NER.

Por ejemplo, en la Figura 8 se muestra un texto de entrada y su salida, donde las entidades reconocidas están marcadas con etiquetas NER.

Respecto al modelo a usar, se optó por usar el modelo pre-entrenado `es_core_news_lg` de spaCy. Este modelo es uno de los modelos pre-entrenados que vienen con la librería, y es un modelo entrenado en el idioma español.

Aunque *spaCy* permite personalizar la arquitectura del modelo y también los hiper-parámetros a usar durante el entrenamiento del mismo, el modelo y parámetros por defecto serán usados.

Este modelo será entrenado para reconocer las entidades mostradas en la Tabla 2.

Tabla 2 - Entidades a reconocer por el modelo entrenado.

Etiqueta de la entidad	Descripción	Ejemplo
FACULTY	Nombre de la facultad a la que el autor pertenece	*FACULTAD DE INGENIERÍA*
DEPARTMENT	Nombre de la carrera a la que el autor pertenece	*INGENIERÍA MECÁNICA Y ELECTROMECÁNICA*
TITLE	Título del proyecto de grado o tesis	*DISEÑO Y CONSTRUCCIÓN DE PROTOTIPO DE LLAVE AUTOMÁTICA DE TRANSFERENCIA ELÉCTRICA PARA EL CENTRO DE SALUD "SAN JUAN LAZARETO"*
AUTHOR	Nombre del autor o autores del documento	*Frady Gabriel Torrico Sacur*
ADVISOR	Nombre del tutor	*Ciro Alberto Levy Díaz*

Etiqueta de la entidad	Descripción	Ejemplo
YEAR	Año de publicación del documento	*2023*

Fuente: Elaboración propia.

Se escogió SpaCy por sobre otras herramientas por los siguientes motivos:

- Es uno de los *toolkits* de NLP mas ampliamente usado en la industria.
- Esta enfocado en el desarrollo de sistemas NER para consumo final.
- Permite el desarrollo de pipelines NER basados en modelos de ML.
- Tiene una numerosa comunidad de voluntarios que desarrollan integraciones con otras herramientas.
- Genera artefactos que pueden ser desplegados fácilmente en servidores.
- Provee arquitecturas de pipeline pre-definidas, varias de ellas con buen porcentaje de precisión.

3.2.1.2 Subsistema de generación de palabras clave

Este subsistema consistirá en un modelo de LLM pre-entrenado en el idioma español. La entrada del modelo será el texto del resumen de un documento académico, y la salida será una lista de palabras o frases que mejor describan los tópicos del documento.

Respecto al modelo a usar, se optó por usar el modelo pre-entrenado `Mistral-7B-Instruct-v0.1` de Mistral. Este modelo es un modelo de LLM entrenado en el idioma español y ajustado para seguir instrucciones.

Se escogió este modelo por sobre otros por los siguientes motivos:

- Es un modelo de LLM relativamente pequeño (7 mil millones de parámetros), por lo que puede ser ejecutado en una computadora de propósito general.
- Es un modelo de LLM entrenado en el idioma español.
- Es un modelo de LLM ajustado para seguir instrucciones, lo que significa que es un modelo que puede ser entrenado para generar texto en base a instrucciones.
- Es un modelo de LLM que supera a otros modelos de LLM en varios benchmarks.

- Es un modelo de LLM de uso libre, es decir, puede ser descargado y usado sin restricciones.
- Es un modelo de LLM que puede ser Ajustado en una computadora de propósito general.

Se habian considerado otros modelos de LLM, sin embargo fueron descartados por los siguientes motivos:

- **GPT 3.5 Turbo:** Es un modelo de LLM accesible mediante un servicio de suscripción en la nube, por lo que no cumple con el requerimiento de que el sistema no genere gastos recurrentes. Este modelo fue usado para el desarrollo del prototipo inicial del sistema, y por ello se lo usar como referencia para comparar el rendimiento del modelo que se desarrolle.
- **Flan-T5-XL:** Es un modelo ideal para el desarrollo de sistemas de generación de palabras clave, sin embargo no tiene soporte para el idioma español, razón por la que fue descartado.
- **Llama-2-7b-chat-hf:** Modelo muy similar al modelo seleccionado, sin embargo su rendimiento en los benchmarks es inferior al modelo seleccionado.

La tabla Tabla 3 muestra las características de los modelos de LLM considerados para el desarrollo del sistema. En la misma tabla también se muestra a `GPT 3.5 Turbo`, sólo como referencia.

Tabla 3 - Selección de LLM basado en sus carácterísticas

Nombre del LLM	Mistral-7B-Instruct-v0.1	Llama-2-7b-chat-hf	Flan-T5-XL	GPT 3.5 Turbo
Arquitectura	Solo decoder	Solo decoder	Encoder-Decoder	Solo decoder
Número de parámetros (en miles de millones)	7	7	3	175[9]
Fecha de publicación	Sept. 2023	Julio 2023	2022	2022
Licencia	Apache 2.0	Meta	CC by 4	Comercial

[9] Aunque OpenAI no ha revelado el tamaño real de su modelo GPT3.5, se rumora que tiene el mismo tamaño que su versión 3, osea 175 mil millones de parámetros.

Nombre del LLM	Mistral-7B-Instruct-v0.1	Llama-2-7b-chat-hf	Flan-T5-XL	GPT 3.5 Turbo
Soporte del idioma español	Si	Si	No	Si
Ajustado para seguir instrucciones	Si	Si	Si	Si
Rendimiento en MT-Bench	6.84[10]	6.27[11]		8.39[12]
Rendimiento en MMLU	60.1[13]	44.4[14]	52.4[15]	70[16]
Velocidad de inferencia (200 tokens)	~2s	~2s	~10s	~2s

Fuente: Elaboración propia.

Los criterios en detalle para la selección del modelo de LLM fueron:

- **Arquitectura:** Se prefería una arquitectura decodificador-codificador, ya que este tipo de modelos son ideales para la síntesis de texto. Sin embargo, no se encontró ningún LLM decodificador-codificador con soporte completo en español.
- **Tamaño:** Se consideraron solo LLMs relativamente pequeños (de 7 mil millones de parámetros o menos). Esto se debe a que solo estos modelos (cuantizados a 4 bits) pueden ejecutarse en computadoras de propósito general.
- **Licencia:** El LLM debía tener una licencia permisiva que permita a la Universidad entrenarlo, instalarlo en un servidor privado y usarlo con fines comerciales.
- **Idioma:** El LLM debía estar entrenado en un corpus de texto en español para procesar

[10] Fuente: Jiang et al. (2023), sección Instruction Finetuning
[11] Fuente: Jiang et al. (2023), sección Instruction Finetuning
[12] Fuente: LMSYS (2023)
[13] Fuente: Jiang et al. (2023), sección Results
[14] Fuente: Jiang et al. (2023), sección Results
[15] Fuente: Chung et al. (2022), sección Putting it all together
[16] Fuente: LMSYS (2023)

efectivamente el texto del resumen de los documentos.

- **Sensibilidad a las instrucciones:** El LLM debe estar ajustado para seguir instrucciones de manera que pueda aplicarse Ingeniería de Prompting en el mismo.

Dos de los mejores LLMs que cumplían con estos criterios son `Mistral-7B-Instruct` y `Llama-2-7b-chat-hf`. Ambos modelos han demostrado un excelente rendimiento en pruebas estándar para LLMs (por ejemplo las pruebas `MT-Bench` y `MMLU`).

Mistral-7B-Instruct tiene un rendimiento ligeramente superior a Llama-2-7b-chat-hf. Además, Mistral-7B-Instruct está disponible en una versión cuantizada de 4 bits, lo que lo hace más eficiente para su ejecución en una computadora de propósito general.

Basándose en los criterios mencionados anteriormente, se seleccionó `Mistral-7B-Instruct` para el proyecto.

De aquí en adelante el resto de las etapas se describen en forma independiente para cada subsistema.

3.3 Subsistema de extracción de metadatos

Como se mencionó anteriormente, el objetivo de este subsistema es extraer los metadatos hallados en la carátula de los documentos académicos.

A continuación se describen las etapas restantes del proyecto para este subsistema.

3.3.1 Etapa 3: Requisitos de datos

¿Que datos se necesitan para entrenar el modelo?

Para entrenar el modelo se necesitan ejemplos de textos de carátulas anotados con las entidades a reconocer. Por ejemplo, en la Figura 15 se muestra un texto de carátula anotado con las entidades a reconocer.

¿Desde donde se deben obtener estos ejemplos?

Los datos con los cuales construir los ejemplos de textos de carátulas anotadas deben ser obtenidos desde el repositorio de documentos académicos de la UMSA, pues es allí donde se encuentran miles de documentos académicos que han sido registrados manualmente por el personal administrativo de

la biblioteca. Ya que el objetivo del proyecto es automatizar este proceso, los documentos registrados manualmente son los mejores ejemplos para entrenar el modelo.

¿Cuales son los criterios para seleccionar estos datos?

Los criterios para seleccionar los datos son:

- Los datos de ejemplo deben provenir de documentos académicos de fin de grado. Esto es, documentos de tipo Proyecto de Grado, Tesis de Grado, Tesis de Maestría y Memorias Laborales. Esto se debe a que los datos de ejemplo deben ser similares a los datos que se procesarán en producción. Otros tipos de documentos académicos como ser artículos de investigación, no son similares a los documentos de fin de grado, ni en su presentación ni en su contenido por lo que no son buenos ejemplos para entrenar el modelo.
- Los datos de ejemplo deben provenir de distintas o todas las facultades de la UMSA. Esto es, los datos de ejemplo deben ser representativos de los datos que se procesarán en producción. Si los datos de ejemplo provienen de una sola facultad, el modelo podría no generalizar bien para los documentos de otras facultades.

¿En que formato deben estar los datos de entrenamiento?

Los datos de entrenamiento deben estar en un formato adecuado para el entrenamiento del modelo NER en SpaCy. Este formato consiste en un archivo binario de extensión `.spacy`. El mismo contiene un conjunto de objetos `Doc`, uno por cada ejemplo de texto de carátula anotado. Cada objeto `Doc` contiene el texto de la carátula y las entidades anotadas.

Este formato de archivo es descrito en la Sección 2.4.4.

¿Que cantidad de datos se necesitan?

Según Mattingly (2020), se necesitan al menos 6000 ejemplos de textos de carátulas anotadas con las entidades a reconocer. Esto se debe a que los modelos de ML basados en CNNs, como es el caso del modelo usado, requieren de una gran cantidad de datos para entrenar. Si se cuenta con menos de 6000 ejemplos, el modelo podría no poder aprender a reconocer las entidades lo suficientemente bien.

3.3.2 Etapa 4: Recopilación de datos

¿Existen los datos requeridos?

Aunque no se cuenta con ejemplos de textos de carátulas anotadas, los mismos se pueden obtener a partir de los metadatos de los documentos académicos y los textos de las carátulas. Asi pues, tras un proceso de ingeniería de características, se pueden obtener textos de carátulas anotados con las entidades a reconocer.

Por tanto, los datos requeridos para construir los datos de entrenamiento son:

- Textos de carátulas
- Metadatos de los documentos académicos

Estos datos se encuentran disponibles en el repositorio de documentos académicos de la UMSA. Los textos de las carátulas se encuentran en los archivos PDF, y los metadatos se encuentran en archivos XML adjuntos a cada documento. Ambos tipos de archivos se encuentran disponibles para su descarga. Existen mas de 30000 documentos académicos en el repositorio, por lo que se cuenta con una cantidad suficiente de datos para entrenar el modelo.

Asimismo, los documentos académicos se encuentran distribuidos en las 13 facultades de la UMSA, por lo que se cuenta con una buena representación de los datos que se procesarán en producción.

3.3.2.1 Tareas realizadas para la recopilación de datos

Para la recopilación de datos se realizaron las siguientes tareas:

- Descarga de los archivos PDF y XML
- Extracción de los metadatos
- Extracción de los textos de las carátulas
- Consolidación de los datos

A continuación se describe cada una de estas tareas.

3.3.2.1.1 Descarga de los archivos PDF y XML

Los archivos PDF y XML fueron descargados desde el repositorio de documentos académicos de la UMSA. Los archivos PDF fueron descargados en forma masiva mediante el uso de un script de Python

usando la librería de scraping `Scrapy`. Esto permitió descargar los archivos en un tiempo razonable.

3.3.2.1.2 Extracción de los metadatos

Los metadatos fueron extraídos desde los archivos XML. Esto se hizo mediante el uso de un script de Python usando la librería `xml`.

La Figura 18 muestra el ejemplo de uno de los archivos XML procesados y el resultado de la extracción de los metadatos.

Figura 18 - Ejemplo de extracción de datos mediante *parsing* del archivo XML.

(a) Archivo XML de Metadatos


```xml
<?xml version="1.0" encoding="UTF-8"?>
...
<dim:dim dspaceType="ITEM">
    <dim:field qualifier="advisor" mdschema="dc" element="contributor">Amusquivar Caballero, Ahmed,
    tutor</dim:field>
    <dim:field qualifier="author" mdschema="dc" element="contributor">Bautista Flores, Carolina</dim:field>
    <dim:field qualifier="accessioned" mdschema="dc" element="date">2023-07-04T15:34:28Z</dim:field>
    <dim:field qualifier="available" mdschema="dc" element="date">2023-07-04T15:34:28Z</dim:field>
    <dim:field qualifier="issued" mdschema="dc" element="date">2020</dim:field>
    <dim:field qualifier="uri" mdschema="dc"
    element="identifier">http://repositorio.umsa.bo/xmlui/handle/123456789/32328</dim:field>
    <dim:field qualifier="abstract" mdschema="dc" language="es_ES" element="description">El objetivo del presente
    proyecto es diseñar un sistema de mejora productiva basándose en el pensamiento esbelto, mediante
    herramientas de la Manufactura Esbelta, para una curtiembre en la ciudad de El Alto, que genere un aumento de
    la productividad global y una reducción de costos. Se realizó un diagnóstico mediante observación directa en
    planta de producción hacia los métodos de trabajo y operarios encargados de los procesos productivos en
    planta, identificándose problemas como desperdicios y las causas de la baja productividad. También se realizó
    un estudio de tiempos para una planificación productiva que vaya de acuerdo al ritmo de pedido del cliente,
    además se diseño un balanceo de línea para asignar la cantidad de operarios requeridos en cada estación de
    trabajo, se desarrollo un sistema de flujo continuo que conlleva a la producción con un sistema pull mediante
    la aplicación del kanban y la implementación de un supermercado de productos terminados, junto con una caja
    heijunka que permite la flexibilidad que se requiere para ser una empresa competitiva que responda a la
    variabilidad de la demanda del cliente. Se planteó una redistribución de planta para la maquinaria y
    estaciones de trabajo acorde al planteamiento del diseño de mejora productiva, también se reduce el lote de
    producción que conlleva a la reducción de otros desperdicios como inventarios entre procesos y espera de
    materiales en estaciones de trabajo.</dim:field>
    <dim:field qualifier="iso" mdschema="dc" language="es_ES" element="language">es</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="subject">PROCESO DE PRODUCCIÓN</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="subject">CURTIEMBRE REMO</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="subject">SISTEMA DE MEJORA PRODUCTIVA</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="subject">MANUFACTURA ESBELTA</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="title">Diseño de de un sistema de mejora productiva
    mediante herramientas de la manufactura esbelta en la Curtiembre REMO E.U.</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="type">Thesis</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="thesisdegreegrantor">Universidad Mayor de San Andrés.
    Facultad de Ingeniería. Carrera de Ingeniería Industrial</dim:field>
    <dim:field mdschema="dc" language="es_ES" element="thesisdegreename">Licenciatura en Ingeniería
    Industrial</dim:field>
</dim:dim>
...
```


(b) Datos extraídos


```json
{
    "palabras_clave": "PROCESO DE PRODUCCIÓN|CURTIEMBRE REMO|SISTEMA DE MEJORA PRODUCTIVA|MANUFACTURA ESBELTA",
    "tutores": "Amusquivar Caballero, Ahmed",
    "autores": "Bautista Flores, Carolina",
    "fecha": "2020",
    "resumen": "El objetivo del presente proyecto es diseñar un sistema de mejora productiva basándose en el
    pensamiento esbelto, mediante herramientas de la Manufactura Esbelta, para una curtiembre en la ciudad de El
    Alto, que genere un aumento de la productividad global y una reducción de costos. Se realizó un diagnóstico
    mediante observación directa en planta de producción hacia los métodos de trabajo y operarios encargados de
    los procesos productivos en planta, identificándose problemas como desperdicios y las causas de la baja
    productividad. También se realizó un estudio de tiempos para una planificación productiva que vaya de acuerdo
    al ritmo de pedido del cliente, además se diseño un balanceo de línea para asignar la cantidad de operarios
    requeridos en cada estación de trabajo, se desarrollo un sistema de flujo continuo que conlleva a la
    producción con un sistema pull mediante la aplicación del kanban y la implementación de un supermercado de
    productos terminados, junto con una caja heijunka que permite la flexibilidad que se requiere para ser una
    empresa competitiva que responda a la variabilidad de la demanda del cliente. Se planteó una redistribución
    de planta para la maquinaria y estaciones de trabajo acorde al planteamiento del diseño de mejora productiva,
    también se reduce el lote de producción que conlleva a la reducción de otros desperdicios como inventarios
    entre procesos y espera de materiales en estaciones de trabajo.",
    "otorgante": "Universidad Mayor de San Andrés. Facultad de Ingeniería. Carrera de Ingeniería Industrial",
    "titulo": "Diseño de de un sistema de mejora productiva mediante herramientas de la manufactura esbelta en la
    Curtiembre REMO E.U."
}
```


Fuente: Elaboración propia.

El resultado del proceso de extracción de los metadatos fue un Pandas Dataframe (ver Figura 19) con los siguientes campos:

- `id`: Identificador único del documento académico.
- `faculty`: Facultad a la que pertenece el documento académico.
- `department`: Carrera a la que pertenece el documento académico.
- `title`: Título del documento académico.
- `author`: Nombre del autor o autores del documento académico.
- `advisor`: Nombre del tutor o tutores del documento académico.
- `year`: Año de publicación del documento académico.
- `type`: Tipo de documento académico.
- `url`: URL del documento académico.

Figura 19 - Pandas Dataframe resultante del parsing realizado desde los archivos XML de metadatos.

1-10 16098 rows × 5 columns pd.DataFrame CSV

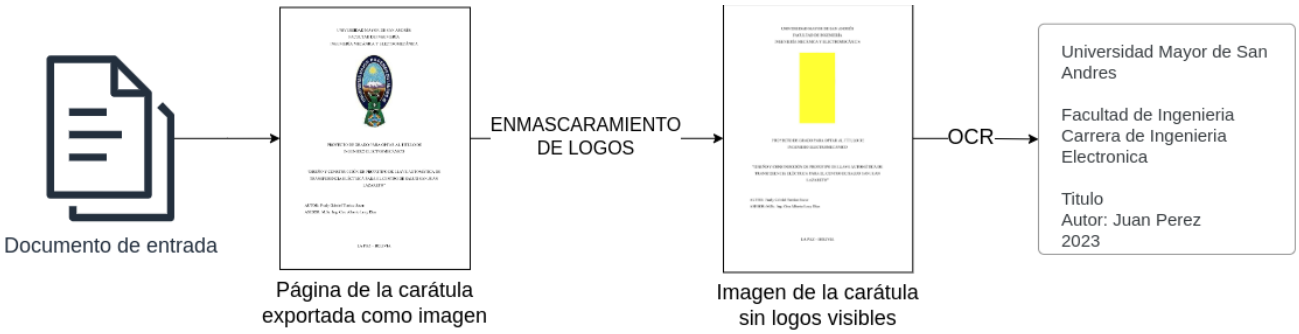
	faculty	title	authors	advisors	year
3	Facultad de Tecnología	Optimización del sistema…	Cuareti Condori, Nelson Lucio	Marquez Gonzales, Luis Richard, tutor	2022
4	Facultad de Tecnología	Desarrollo de un sistema…	Marquez Castro, Vannia Estephan…	Marin Thames, José Arturo, tutor	2021
5	Facultad de Tecnología	Diseño e implementación …	Ibañez Durán, Valeria Sonia	Marin Thames, José Arturo, tutor	2021
6	Facultad de Arquitectura, Ar…	Escuela de idiomas y dif…	Velasco Aguilar, Denisse Andrea	Sainz Cardona, Jorge, asesor	2021
8	Facultad de Tecnología	Diseño del proyecto de a…	Condori Cuenca, Rabi Ruben	Tiñini, Oswaldo, tutor	2019
9	Facultad de Tecnología	Rediseño del sistema de …	Quispe Ramirez, Edgardo	Mamani Villca, Néstor, tutor	2022
10	Facultad de Arquitectura, Ar…	Bloque de concreto estru…	Mamani Cortez, Dayane Esther	Alvarado Mollinedo, Belén, asesora	2020
11	Facultad de Tecnología	Diseño de un horno de in…	Arcani Cahuaya, Rene Bladimir	Mamani Villca, Nestor, tutor	2022
12	Facultad de Ciencias Puras y…	Plataforma de empleo vir…	Nigañez Aliaga, Juan Felipe	Flores Morales, Rosa, tutor	2021

Fuente: Elaboración propia.

3.3.2.1.3 Extracción de los textos de las carátulas

Para extraer el texto de las carátulas se desarrolló un pipeline de procesamiento de imágenes basado en la librería `OpenCV`. La Figura 20 muestra una representación gráfica del pipeline.

Figura 20 - Extracción del texto de la carátula



Fuente: Elaboración propia.

El pipeline consistió en los siguientes pasos:

1. **Preprocesamiento de la imagen:** Se aplicaron técnicas de procesamiento de imágenes para mejorar la calidad de la imagen y facilitar la extracción del texto. Estas técnicas incluyeron: binarización, dilatación, erosión, enmascaramiento de regiones de interés.

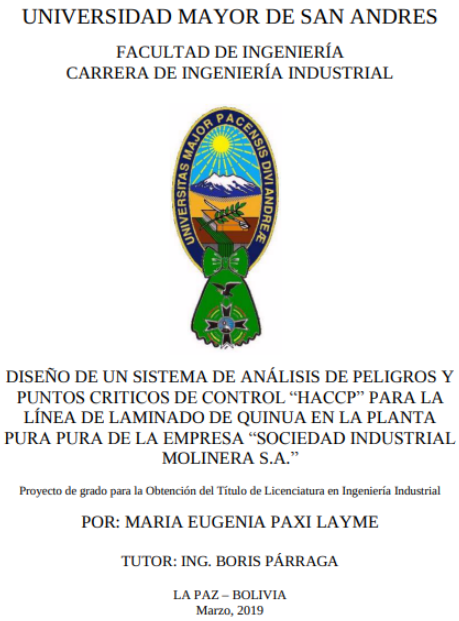
2. **Extracción del texto:** Se usó el software de reconocimiento óptico de caracteres (OCR por sus siglas en inglés) `Tesseract` para extraer el texto de la carátula.

3. **Postprocesamiento del texto:** Se aplicaron técnicas de procesamiento de texto para mejorar la calidad del texto extraído. Estas técnicas incluyeron: eliminación de espacios en blanco innecesarios, eliminación de caracteres no ASCII, eliminación de caracteres inválidos y eliminación de líneas vacías.

Un ejemplo de la extracción del texto de una carátula es mostrado en la Figura 21.

Figura 21 - Ejemplo de la extracción del texto desde una carátula.

(a) Carátula


UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA INDUSTRIAL

DISEÑO DE UN SISTEMA DE ANÁLISIS DE PELIGROS Y PUNTOS CRITICOS DE CONTROL "HACCP" PARA LA LÍNEA DE LAMINADO DE QUINUA EN LA PLANTA PURA PURA DE LA EMPRESA "SOCIEDAD INDUSTRIAL MOLINERA S.A."

Proyecto de grado para la Obtención del Título de Licenciatura en Ingeniería Industrial

POR: MARIA EUGENIA PAXI LAYME

TUTOR: ING. BORIS PÁRRAGA

LA PAZ – BOLIVIA
Marzo, 2019

(b) Texto extraído desde la carátula.

UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA INDUSTRIAL

DISENO DE UN SISTEMA DE ANALISIS DE PELIGROS Y
PUNTOS CRITICOS DE CONTROL "HACCP" PARA LA
LINEA DE LAMINADO DE QUINUA EN LA PLANTA
PURA PURA DE LA EMPRESA "SOCIEDAD INDUSTRIAL
MOLINERA S.A."

Proyecto de grado para la Obtencion del Titulo de Licenciatura en Ingenieria Industrial
POR: MARIA EUGENIA PAXI LAYME
TUTOR: ING. BORIS PARRAGA

LA PAZ -- BOLIVIA
Marzo, 2019

Fuente: Elaboración propia.

El resultado de procesar todos los datos se consolidó en un Dataframe de Pandas (ver Figura 22) con los siguientes campos:

- `id`: Identificador único del documento académico.
- `text`: Texto de la carátula del documento académico.

Figura 22 - Pandas Dataframe resultante de la extracción del texto de la carátula de todos los documentos recolectados.

1-10 16098 rows × 1 columns pd.DataFrame CSV

	pdf_cover_page
3	UNIVERSIDAD MAYOR DE SAN ANDRES\n\nFACULTAD DE TECNOLOGÍA\nCARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES\n\nOPTIMIZACIÓN DEL SI…
4	UNIVERSIDAD MAYOR DE SAN ANDRES\nFACULTAD DE TECNOLOGIA\nCARRERA DE ELECTRONICA Y TELECOMUNICACIONES\n\nDESARROLLO DE UN SIST…
5	UNIVERSIDAD MAYOR DE SAN ANDRÉS\n\nFACULTAD DE TECNOLOGÍA\nCARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES\n\n"DISEÑO E IMPLEMENT…
6	UNIVERSIDAD MAYOR DE SAN ANDRES\nFACULTAD DE ARQUITECTURA, ARTES Y DISEÑO Y URBANISMO\nCARRERA DE ARQUITECTURA\n\nPROYECTO DE…
8	UNIVERSIDAD MAYOR DE SAN ANDRÉS\n\nFACULTAD DE TECNOLOGÍA\nCARRERA DE ELECTRICIDAD INDUSTRIAL\n\nDISEÑO DEL PROYECTO DE ALUMB…
9	UNIVERSIDAD MAYOR DE SAN ANDRÉS\nFACULTAD DE TECNOLOGÍA\nCARRERA DE ELECTRICIDAD INDUSTRIAL\n\nPROYECTO DE GRADO A NIVEL LICE…
10	UNIVERSIDAD MAYOR DE SAN ANDRES\nFACULTAD DE ARQUITECTURA, ARTES, DISEÑO Y URBANISMO\n\nCARRERA DE ARQUITECTURA\n\nTESIS DE G…
11	UNIVERSIDAD MAYOR DE SAN ANDRES\nFACULTAD DE TECNOLOGIA\nCARRERA DE ELECTRICIDAD INDUSTRIAL\n\nPROYECTO DE GRADO A NIVEL LICE…
12	UNIVERSIDAD MAYOR DE SAN ANDRÉS\nFACULTAD DE CIENCIAS PURAS Y NATURALES\nCARRERA DE INFORMÁTICA\n\nTESIS DE GRADO\n"PLATAFORM…
13	UNIVERSIDAD MAYOR DE SAN ANDRÉS\nFACULTAD DE TECNOLOGIA\nCARRERA DE GEODESIA, TOPOGRAFÍA Y GEOMÁTICA\n\nINFORME DE TRABAJO DI…

Fuente: Elaboración propia.

3.3.2.1.4 Consolidación de los datos

Los datos de los metadatos y los textos de las carátulas fueron consolidados en un solo Pandas Dataframe (ver Figura 23) con los siguientes campos:

- `id`: Identificador único del documento académico.
- `faculty`: Facultad a la que pertenece el documento académico.
- `department`: Carrera a la que pertenece el documento académico.
- `title`: Título del documento académico.
- `author`: Nombre del autor o autores del documento académico.
- `advisor`: Nombre del tutor o tutores del documento académico.
- `year`: Año de publicación del documento académico.
- `type`: Tipo de documento académico.
- `url`: URL del documento académico.
- `cover_page_text`: Texto de la carátula del documento académico.

3.3.3 Etapa 5: Comprensión de datos

El análisis de los datos reveló las siguientes características:

Figura 23 - Pandas Dataframe resultante de la consolidación de los datos de los metadatos y los textos de las carátulas.

1-10 16098 rows × 8 columns pd.DataFrame CSV

	pdf_cover_page	faculty	title	authors	advisors	subjects	abstract	year
3	UNIVERSIDAD MAYO…	Facultad de Te…	Optimizaci…	Cuareti Condo…	Marquez Gon…	SISTEMA DE SEGUR…	Nuestro país se encuentra en vía…	2022
4	UNIVERSIDAD MAYO…	Facultad de Te…	Desarrollo…	Marquez Castr…	Marin Thame…	DETECTOR DE INCE…	Este documento trata de la maner…	2021
5	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño e i…	Ibañez Durán,…	Marin Thame…	MEDICIÓN ELECTRÓ…	El presente proyecto surge con l…	2021
6	UNIVERSIDAD MAYO…	Facultad de Ar…	Escuela de…	Velasco Aguil…	Sainz Cardo…	OBRAJES (LA PAZ,…	La propuesta de la "Escuela de I…	2021
8	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño del…	Condori Cuenc…	Tiñini, Osw…	ALUMBRADO PÚBLICO	El alumbrado público es un servi…	2019
9	UNIVERSIDAD MAYO…	Facultad de Te…	Rediseño d…	Quispe Ramire…	Mamani Vill…	PUESTA A TIERRA	Este proyecto está predispuesto …	2022
10	UNIVERSIDAD MAYO…	Facultad de Ar…	Bloque de …	Mamani Cortez…	Alvarado Mo…	BLOQUES DE CONCR…	La necesidad de esta investigaci…	2020
11	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño de …	Arcani Cahuay…	Mamani Vill…	HORNO DE INDUCCI…	El proyecto en general se encuen…	2022
12	UNIVERSIDAD MAYO…	Facultad de Ci…	Plataforma…	Nigañez Aliag…	Flores Mora…	ANÁLISIS Y DISEÑ…	El proyecto explica en el desar…	2021

Fuente: Elaboración propia.

- **Distribución de publicaciones por año:** El análisis de los metadatos contenidos en los archivos XML reveló que cerca del 80% de los documentos fueron publicados a partir del año 2010 (ver Figura 24).
- **Tipos de documento:** Se encontró una gran variedad de tipos de documentos, incluyendo documentos académicos, como tesis, proyectos de grado y trabajos dirigidos, así como artículos de revistas y periódicos. La Figura 25 muestra la distribución de los documentos por tipo.
- **Formato de las carátulas:** Algunos documentos no contaban con una carátula convencional de un documento académico. Estos documentos, como revistas y artículos de periódicos, presentaban un formato diferente. Un par de ejemplos son mostrados en la Figura 27.

En base a estas características se planteo las siguientes preguntas:

- ¿Son los datos representativos de los datos que se procesarán en producción?

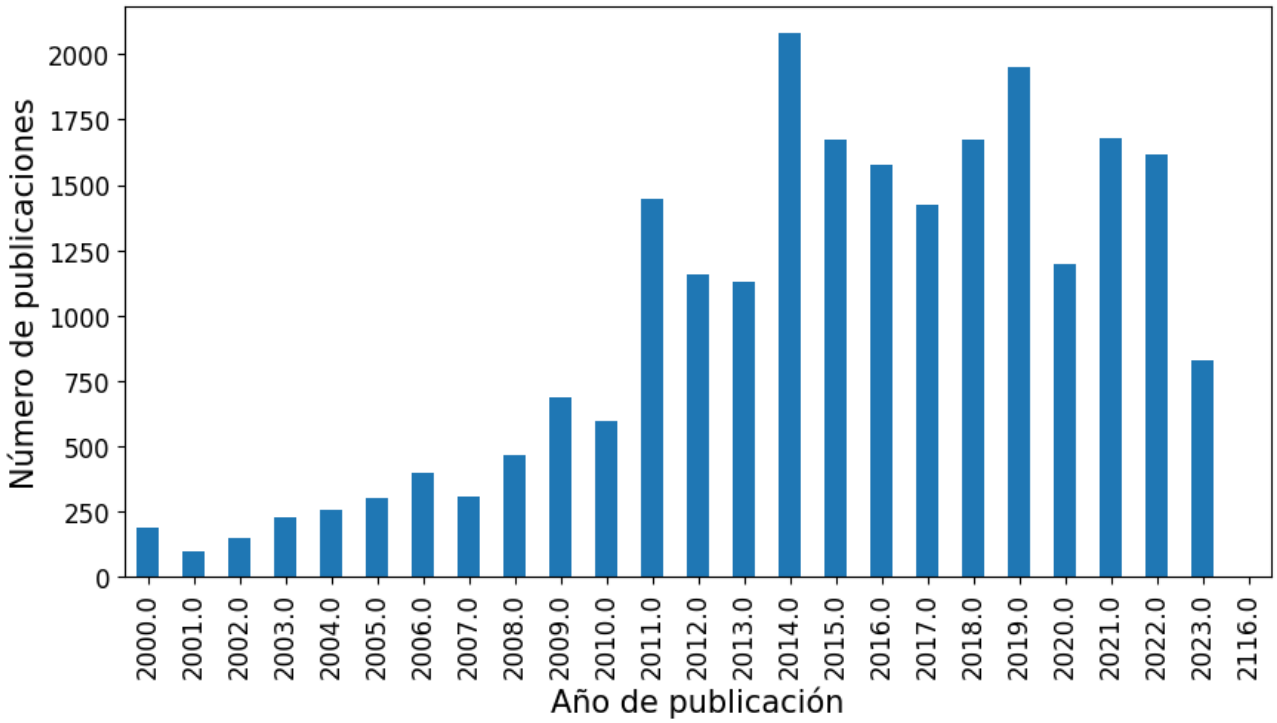
  Si. Los datos representan una buena muestra de los datos que se procesarán en producción.

- ¿Son los datos de buena calidad?

  En general los datos son de buena calidad. Sin embargo, se encontraron algunos documentos que no son de fin de grado, como ser artículos de revistas y periódicos. Estos documentos no son buenos ejemplos para entrenar el modelo, por lo que se decidió descartarlos del conjunto de datos de entrenamiento.

  Asimismo, se encontraron algunos documentos que no contaban con una carátula convencional

Figura 24 - Número de publicaciones por año



Fuente: Elaboración propia.

de un documento académico. Estos documentos tampoco son buenos ejemplos para entrenar el modelo, por lo que también se decidió descartarlos del conjunto de datos de entrenamiento.

Tambien se encontraron algunos documentos muy antiguos, como ser documentos de antes del año 2010. Estos documentos tampoco son buenos ejemplos para entrenar el modelo, por lo que también se decidió descartarlos del conjunto de datos de entrenamiento.

Finalmente, se encontraron algunos documentos que no contaban con metadatos, o que los metadatos no correspondían con el documento.

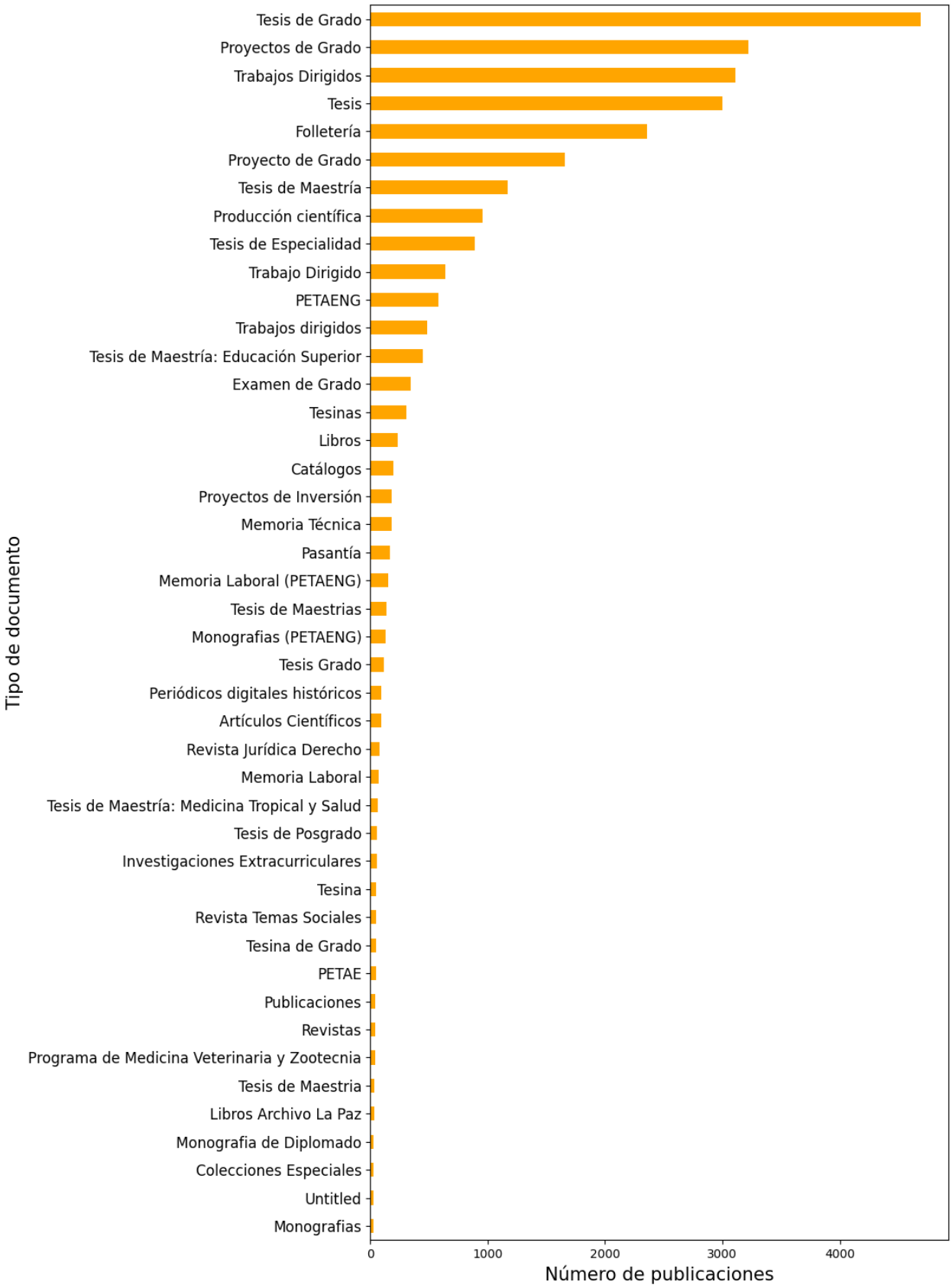
- ¿Harán falta mas datos?

  No. Se cuenta con una cantidad suficiente de datos para entrenar el modelo.

- ¿Que años de publicación se deben incluir en el conjunto de datos de entrenamiento?

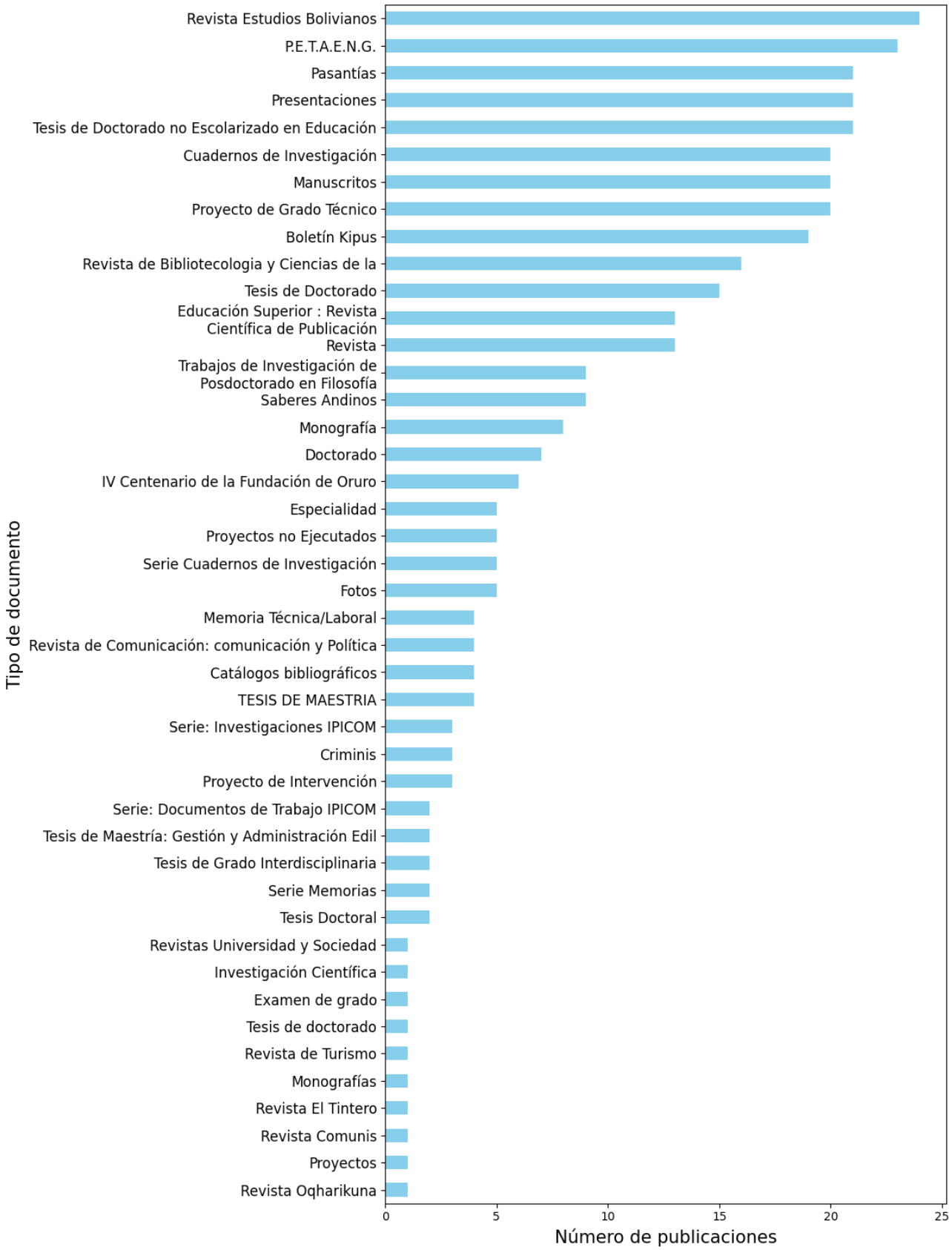
  Se deben incluir los documentos publicados a partir del año 2010. Esto se debe a que cerca del 80% de los documentos fueron publicados a partir de ese año. Incluir los documentos publicados antes del año 2010 no aportaría mucho al modelo, ya que son pocos los documentos

Figura 25 - Número de publicaciones por tipo (primera parte)



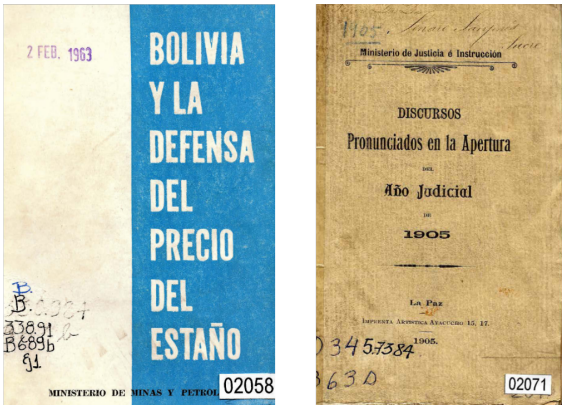
Fuente: Elaboración propia.

Figura 26 - Número de publicaciones por tipo (continuación)



Fuente: Elaboración propia.

Figura 27 - Ejemplo de documentos a descartar del conjunto inicial de documentos. Estos son descartados por no tener un formato de carátula convencional.



Fuente: Elaboración propia.

publicados antes de ese año y aun peor, los documentos publicados antes de ese año podrían tener un formato de carátula diferente al de los documentos publicados después de ese año por ser los primeros documentos académicos de la UMSA.

- ¿Que tipos de documentos se deben incluir en el conjunto de datos de entrenamiento?

  Se deben incluir los documentos de fin de grado, como ser: Proyectos de Grado, Tesis de Grado, Tesis de Maestría y Memorias Laborales. Esto se debe a que los documentos de fin de grado son los documentos que se procesarán en producción.

- ¿Que documentos se deben descartar del conjunto de datos de entrenamiento?

  Se deben descartar los documentos que no son de fin de grado, como ser artículos de revistas y periódicos. Asimismo, se deben descartar los documentos que no cuentan con una carátula convencional de un documento académico. Finalmente, se deben descartar los documentos muy antiguos, como ser documentos de antes del año 2010.

- ¿Qué tareas de limpieza de datos se deben realizar?

  En base a las preguntas anteriores, se deben realizar las siguientes tareas de limpieza de datos:

  - Descartar los documentos que no son de fin de grado, como ser artículos de revistas y periódicos.
  - Descartar los documentos que no cuentan con una carátula convencional de un documento académico.

  - Descartar los documentos muy antiguos, como ser documentos de antes del año 2010.
  - Descartar los documentos que no cuentan con metadatos, o que los metadatos no corresponden con el documento.

- ¿Qué tareas de preparación de datos se deben realizar para que los datos estén listos para el entrenamiento del modelo?

  Tras las tareas de limpieza de datos, se deben realizar las siguientes tareas de preparación de datos:

  - Usar los datos de los metadatos y los textos de las carátulas para contruir el conjunto de datos de entrenamiento. El formato en el cual se debe construir el conjunto de datos de entrenamiento tal como se mencionó anteriormente, es un archivo binario de extensión `.spacy` (ver la sección del marco teórico Sección 2.4.3.1 para mas detalles).

3.3.4 Etapa 6: Preparación de datos

3.3.4.1 Filtrado de datos

En base a las preguntas planteadas en la etapa anterior, se decidió excluir documentos que:

- **Daten de gestiones anteriores a 2010:** Estos documentos podrían haber sido elaborados con poca estandarización en cuanto al formato de las carátulas, y que el software usado para generar los archivos PDF podría haber sido de muy variada calidad, especialmente antes de que MS Office introdujera el soporte para la exportación de PDFs en su versión 2007. Para evitar que estos factores pudieran perjudicar la generalización de los modelos, se decidió descartar estos documentos. Esto redujo el conjunto de datos a 20,031 instancias, que aún eran suficientes para entrenar los modelos.

- **No tengan carátulas convencionales:** Debido a que el propósito del proyecto es extraer metadatos desde documentos de fin de grado, se seleccionaron aquellos que contaban con una carátula convencional propio de este tipo de documentos. Se seleccionaron solo 6 tipos de documentos que cumplen este criterio:

  - Proyectos de grado

  - Tesis de grado

- Trabajos dirigidos
- Tesis de especialidad
- Tesis de maestría
- Proyectos PETAENG

La figura Figura 28 muestra la primera página de un documento aleatorio para cada tipo de documento seleccionado, evidenciando que en efecto estos documentos contienen una carátula convencional. Todos los demás tipos de documentos fueron excluidos.

Tras cada proceso de filtrado, los datos se redujeron en cantidad según el detalle mostrado en la Tabla 4, quedando 16098 documentos resultantes, listos para ser convertidos en los datos de entrenamiento.

Tabla 4 - Criterios de selección de datos y número de datos tras cada operación de filtrado.

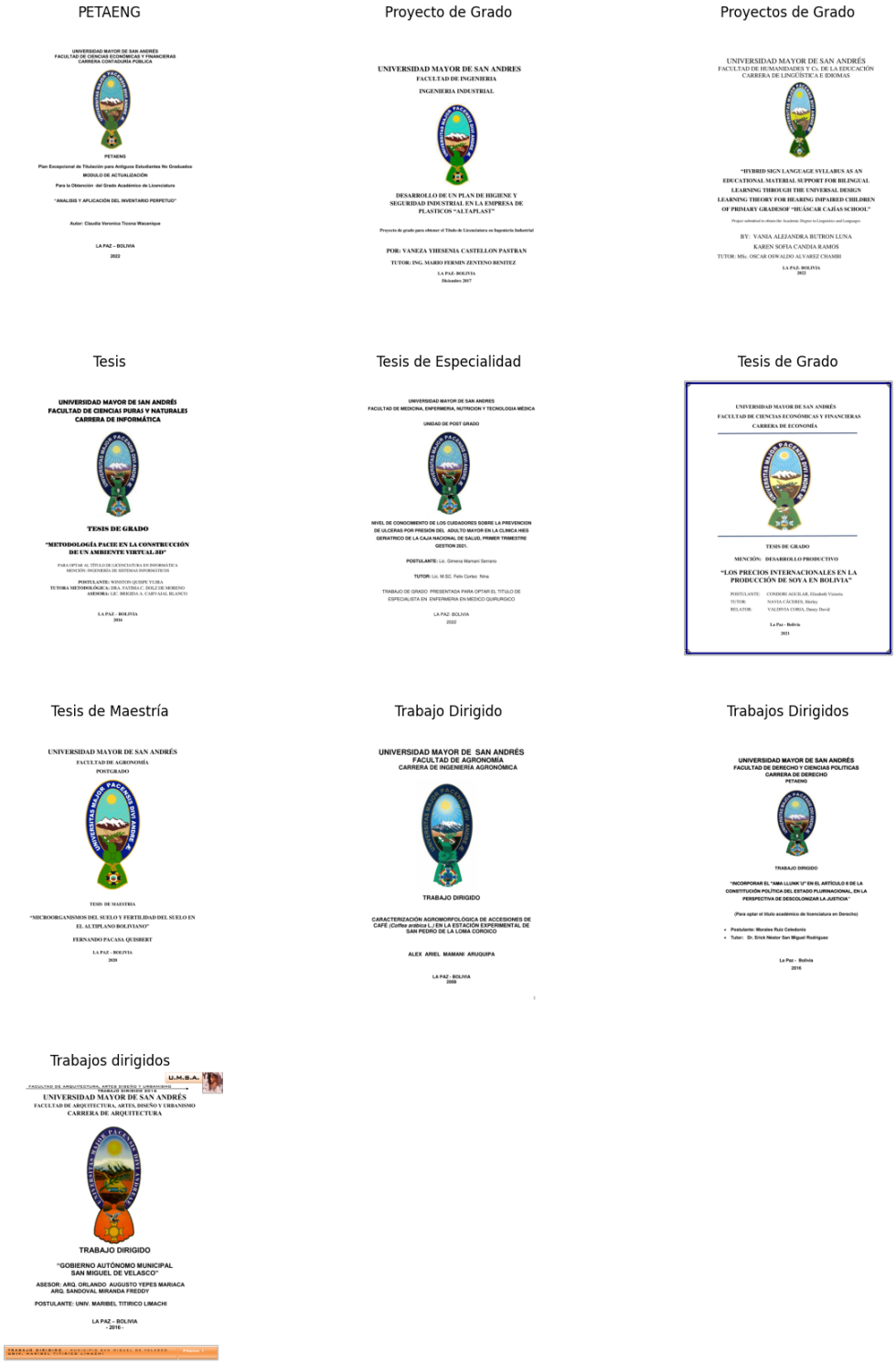
	Número de datos restantes para el entrenamiento
Cantidad inicial de documentos	26837
Filtro 1:Documentos publicados el año 2010 o después	20031
Filtro 2: Documentos de los tipos Proyectos de Grado, Tesis de Grado, Tesis, Trabajos Dirigidos, Tesis de Especialidad, Tesis de Maestría, PETAENG	16098
TOTAL DOCUMENTOS A PROCESAR	16098

Fuente: Elaboración propia.

Como dato adicional, en las figuras Figura 29 y Figura 30 se muestran la cantidad de documentos por facultad y por tipo en el conjunto de datos resultante.

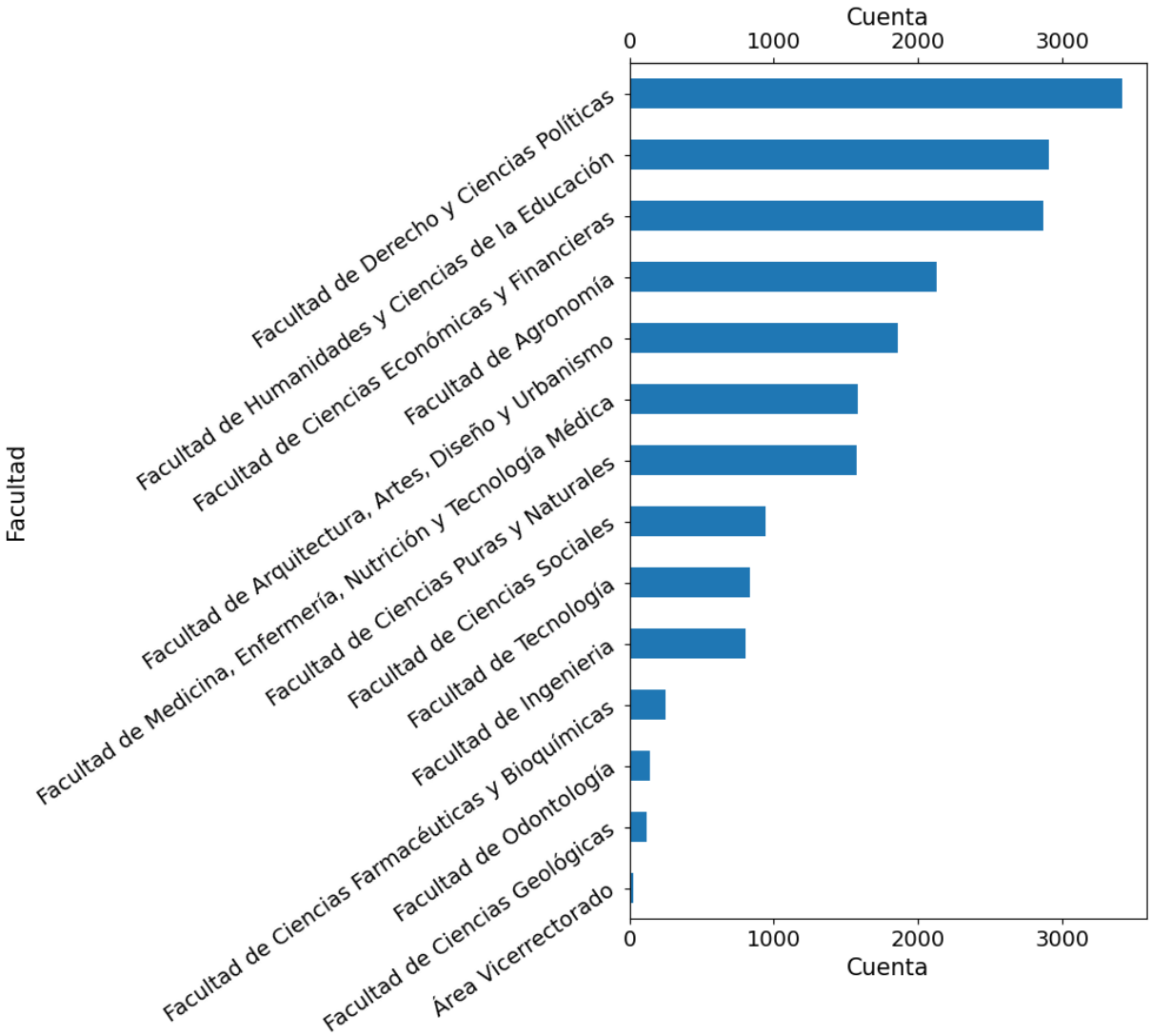
3.3.4.2 Limpieza de datos

Figura 28 - Primera página de una muestra aleatoria de cada tipo de documento seleccionado



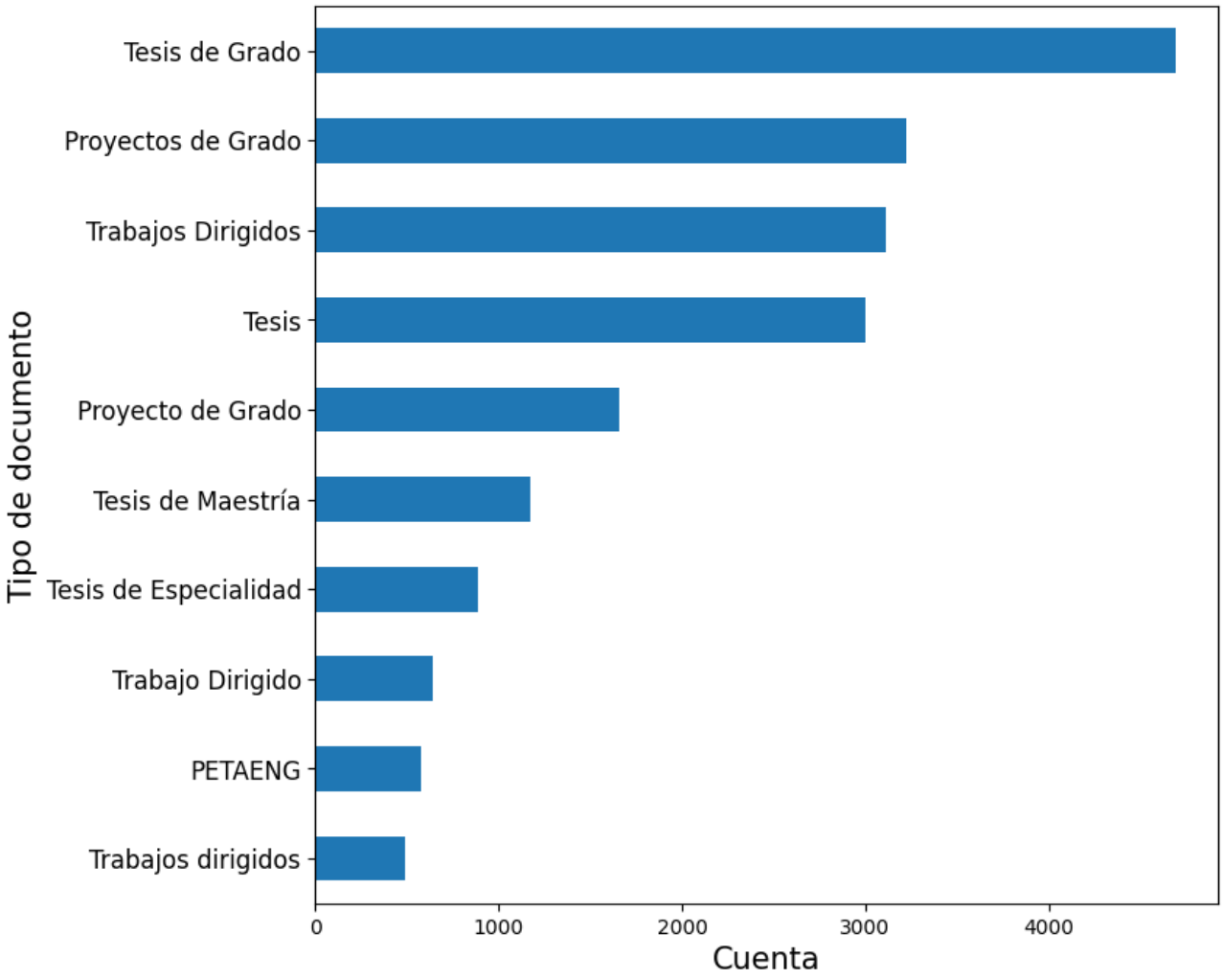
Fuente: Elaboración propia.

Figura 29 - Número de documentos por facultad, en el conjunto de datos final



Fuente: Elaboración propia.

Figura 30 - Número de documentos por tipo, en el conjunto de datos final



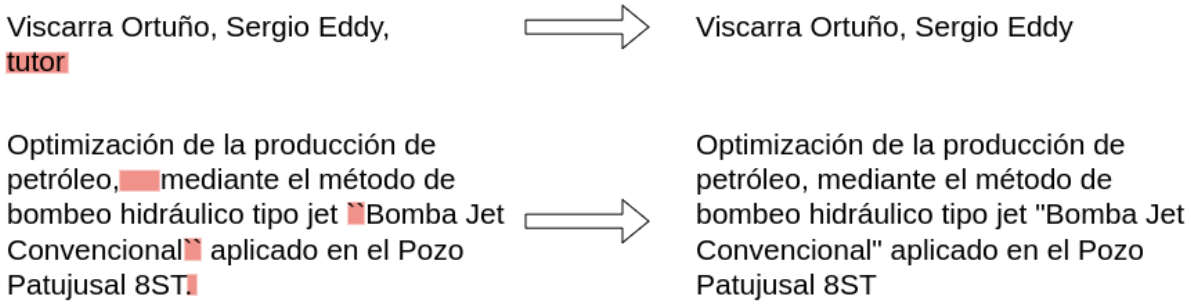
Fuente: Elaboración propia.

Se tuvo que realizar tareas de limpieza de los datos de ambos Dataframes obtenidos en la Sección 3.3.2.1.3 y la Sección 3.3.2.1.2. Esta práctica fue vital para asegurar un buen desempeño del pipeline entrenado resultante. Para ello, mediante un Cuadernos Jupyter se procedió a:

1. Limpiar los textos extraídos de las carátulas:
    1. Reemplazar múltiples espacios en blanco por uno solo.
    2. Convertir los caracteres no ASCII a caracteres ASCII.
2. Limpiar los textos extraídos desde los archivos de metadatos
    1. Remover comillas innecesarias.
    2. Remover puntos finales innecesarios.
    3. Remover palabras que no debían ser parte de los nombres.

Algunos ejemplos de este proceso de limpieza son mostrados en la Figura 31.

Figura 31 - Ejemplos del proceso de limpieza del texto



Fuente: Elaboración propia.

Finalmente ambos Dataframes, ya limpiados, fueron consolidados en un solo Dataframe de Pandas.

3.3.4.2.1 Construcción de los datos de entrenamiento

Para construir de manera automática los datos de entrenamiento, se usó un modelo NER auxiliar basado en reglas.

La Figura 33 muestra una representación gráfica del pipeline.

La entrada del pipeline es un texto de carátula. El pipeline aplica un modelo NER auxiliar basado en reglas para identificar los metadatos en el texto de la carátula. El resultado del pipeline es un nuevo texto de carátula con las entidades anotadas.

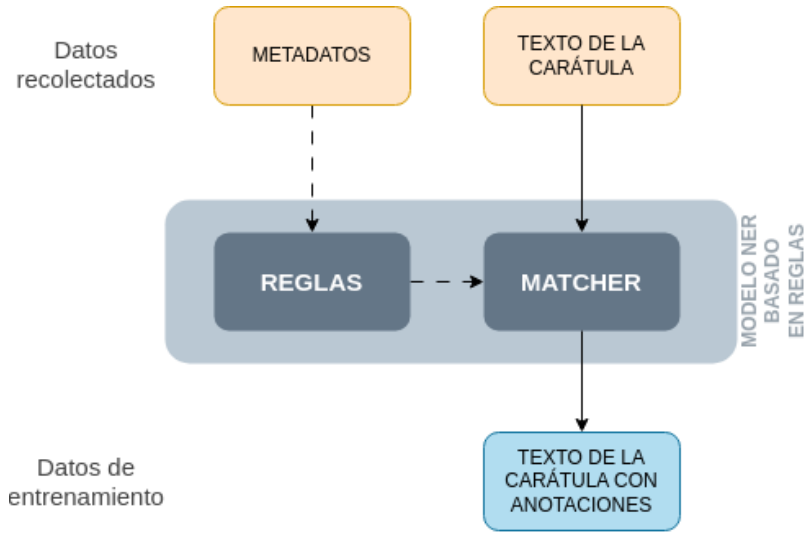
Figura 32 - Dataframe consolidado. El mismo contiene el texto de las carátulas y el texto de cada metadato.

1-10 16098 rows × 8 columns pd.DataFrame CSV

	pdf_cover_page	faculty	title	authors	advisors	subjects	abstract	year
3	UNIVERSIDAD MAYO…	Facultad de Te…	Optimizaci…	Cuareti Condo…	Marquez Gon…	SISTEMA DE SEGUR…	Nuestro país se encuentra en via…	2022
4	UNIVERSIDAD MAYO…	Facultad de Te…	Desarrollo…	Marquez Castr…	Marin Thame…	DETECTOR DE INCE…	Este documento trata de la maner…	2021
5	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño e i…	Ibañez Durán,…	Marin Thame…	MEDICIÓN ELECTRÓ…	El presente proyecto surge con l…	2021
6	UNIVERSIDAD MAYO…	Facultad de Ar…	Escuela de…	Velasco Aguil…	Sainz Cardo…	OBRAJES (LA PAZ,…	La propuesta de la "Escuela de I…	2021
8	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño del…	Condori Cuenc…	Tiñini, Osw…	ALUMBRADO PÚBLICO	El alumbrado público es un servi…	2019
9	UNIVERSIDAD MAYO…	Facultad de Te…	Rediseño d…	Quispe Ramire…	Mamani Vill…	PUESTA A TIERRA	Este proyecto está predispuesto …	2022
10	UNIVERSIDAD MAYO…	Facultad de Ar…	Bloque de …	Mamani Cortez…	Alvarado Mo…	BLOQUES DE CONCR…	La necesidad de esta investigaci…	2020
11	UNIVERSIDAD MAYO…	Facultad de Te…	Diseño de …	Arcani Cahuay…	Mamani Vill…	HORNO DE INDUCCI…	El proyecto en general se encuen…	2022
12	UNIVERSIDAD MAYO…	Facultad de Ci…	Plataforma…	Nigañez Aliag…	Flores Mora…	ANÁLISIS Y DISEÑ…	El proyecto explica en el desar…	2021

Fuente: Elaboración propia.

Figura 33 - Modelo NER basado en reglas usado para generar los datos de entrenamiento



Fuente: Elaboración propia.

Las reglas se construyeron en base a los metadatos obtenidos en la sección anterior. Estas reglas tenían el objetivo de identificar los metadatos en el texto de la carátula correspondiente. Un ejemplo de una regla creada se muestra en la tabla Tabla 5.

Tabla 5 - Ejemplo de reglas definidas en el modelo NER auxiliar. Las reglas fueron basadas en los metadatos extraídos desde los XMLs

Ejemplo de metadato	Ejemplo de regla construída para identificar el metadato en el texto de la carátula

authors: ‘Juan Perez, Luis Romero’	`{`
	`'label': 'AUTHOR',`
	`[`
	`[{'LOWER': 'juan'}],`
	`[{'LOWER'}: 'perez'],`
	`[{'LOWER': 'luis'}],`
	`[{'LOWER'}: 'romero']`
	`]`
	`}`

advisors: ‘Ciro Levy’	`{`
	`'label': 'ADVISOR',`
	`[`
	`[{'LOWER': 'ciro'}],`
	`[{'LOWER': 'levy'}]`
	`]`
	`}`

Fuente: Elaboración propia.

El resultado de aplicar las reglas a cada texto de entrada fue un nuevo texto con anotaciones de los metadatos identificados. De esta forma, se generaron automáticamente un conjunto de datos de entrenamiento con los siguientes campos:

- `text`: Texto de la carátula del documento académico.

- `entities`: Entidades anotadas en el texto de la carátula del documento académico.

Un ejemplo de un texto de carátula anotado es mostrado en la Figura 15.

Finalmente los datos fueron divididos en dos conjuntos: conjunto de entrenamiento y conjunto de prueba. El conjunto de entrenamiento contiene el 80% de los datos, y el conjunto de prueba contiene el 20% de los datos (ver Tabla 6).

Ambos conjuntos de datos fueron guardados en archivos binarios de extensión `.spacy`. Estos archivos son los que se usaron para entrenar el modelo NER.

Tabla 6 - Porcentaje de división de los datos en conjunto de entrenamiento y conjunto de pruebas

Datos de entrenamiento	80%
Datos de prueba	20%

Fuente: Elaboración propia.

3.3.5 Etapa 7: Modelado (1era iteración)

Se entreno el pipeline dejando que Spacy determine automáticamente cuando detener el entrenamiento. El entrenamiento se detuvo cuando la función de pérdida (loss) no mejoró por 5 iteraciones consecutivas. El pipeline fue entrenado con los siguientes parámetros:

- **Tamaño del lote:** 1000
- **Tasa de aprendizaje:** 0.001
- **Tasa de dropout:** 0.1
- **Número maximo de pasos:** 20000
- **Frecuencia de evaluación:** cada 200 pasos

Las metricas de entrenamiento, asi como el modelo entrenado, fueron guardados en un servidor MLFlow.

Al detenerse el entrenamiento el pipeline había alcanzado una *Precision* del 85.77%, un *Recall* del 91.51%, y un F1 del 88.57%. La Figura 35 muestra el valor final de todas las métricas obtenidas tras el entrenamiento.

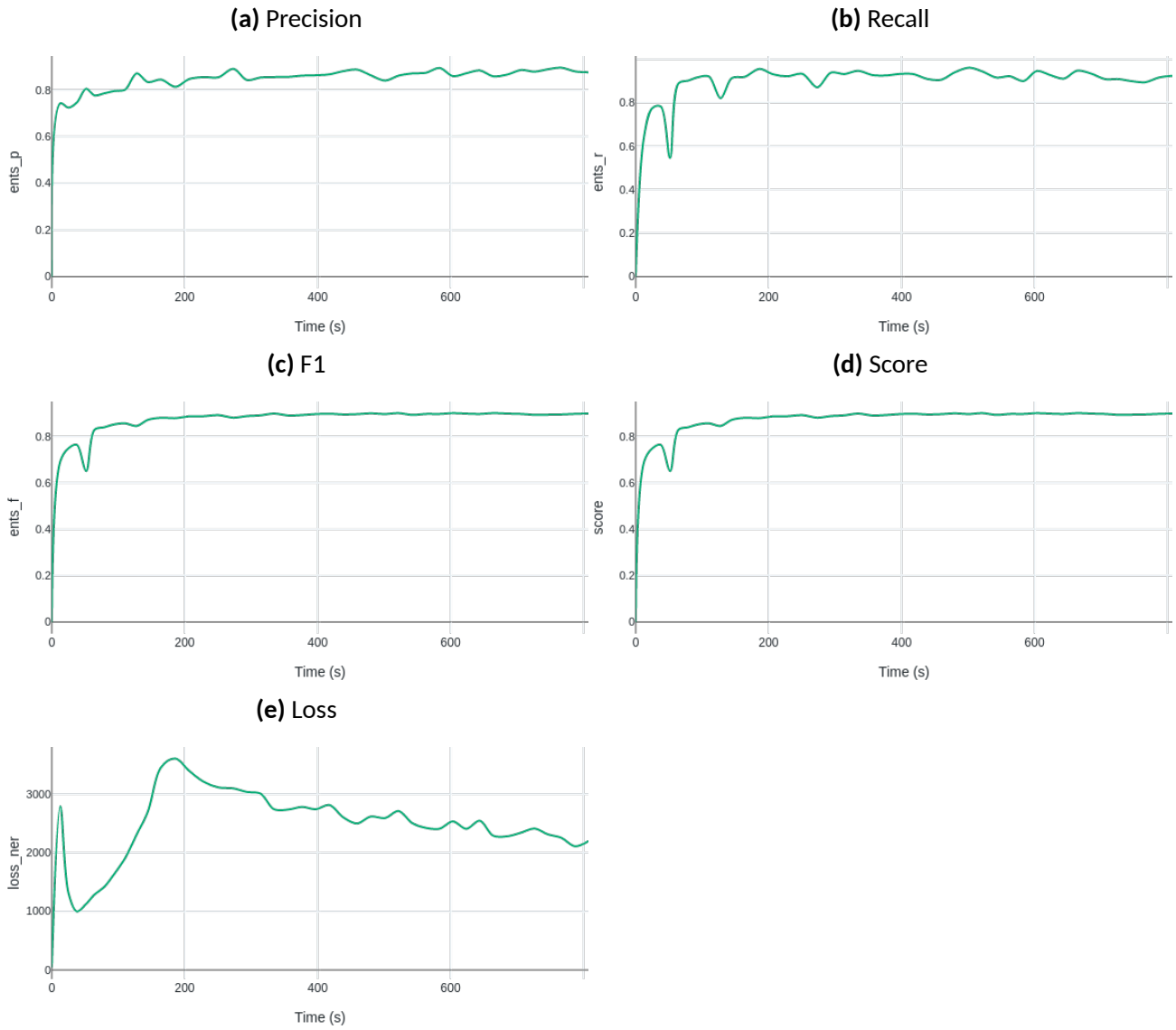
Figura 34 - Estado final de las métricas de entrenamiento. Métricas extraídas desde MLFlow.

Name	Value
ents_f	0.8912809472551131
ents_p	0.8989891426432048
ents_per_type.ADVISOR.f	0.721366097459392
ents_per_type.ADVISOR.p	0.7616534740545294
ents_per_type.ADVISOR.r	0.685126582278481
ents_per_type.AUTHOR.f	0.9474393530997304
ents_per_type.AUTHOR.p	0.9262187088274044
ents_per_type.AUTHOR.r	0.9696551724137931
ents_per_type.DEPARTMENT.f	0.9910250354274918
ents_per_type.DEPARTMENT.p	0.984514312529329
ents_per_type.DEPARTMENT.r	0.997622444127437
ents_per_type.FACULTY.f	0.9866993349667483
ents_per_type.FACULTY.p	0.9744210162461113
ents_per_type.FACULTY.r	0.999291031549096
ents_per_type.TITLE.f	0.5340909090909091
ents_per_type.TITLE.p	0.6244755244755245
ents_per_type.TITLE.r	0.4665621734587252
ents_per_type.YEAR.f	0.9480421686746988
ents_per_type.YEAR.p	0.9230205278592375
ents_per_type.YEAR.r	0.9744582043343654
ents_r	0.8837038127484175
loss_ner	2323.6482324777935
loss_tok2vec	594.816052022298
score	0.8912809472551131
speed	48697.790499794864

Fuente: Elaboración propia.

Asimismo, en la Figura 34 se muestran las gráficas de evolución de las métricas durante el entrenamiento.

Figura 35 - Métricas de entrenamiento



Fuente: Elaboración propia.

3.3.6 Etapa 8: Evaluación (1era iteración)

El análisis de los resultados del entrenamiento, mostrados en las figuras Figura 35 y Figura 34, reveló las siguientes conclusiones:

- La precisión de la detección de la mayoría de las entidades fue alta, superior al 90%. Sin embargo, dos entidades presentaron una precisión baja: ADVISOR (76,17%) y TITLE (62,45%).

- La función de pérdida (loss) indicó que el modelo no pudo converger, lo que podría indicar un problema con los datos de entrenamiento.

Para determinar la causa de los problemas observados, se analizó los datos de entrenamiento. Se encontró que:

- Los documentos más antiguos carecían de etiquetas ADVISOR. Esto se debe a que, en los primeros años de registro de los documentos en el repositorio universitario, no era habitual registrar a los tutores en el momento de la publicación de los documentos.
- Muchos datos de entrenamiento carecían de etiquetas TITLE. Esto se debe a que el modelo basado en reglas no pudo identificar muchos de ellos, debido a que existen mínimas diferencias entre el título contenido en el texto de las carátulas y el texto del título según los archivos de metadatos.

Para mitigar los problemas observados, se decidió construir conjuntos de datos de entrenamiento por tipo de entidad. De esta manera, cada conjunto de datos contendría solo los datos relevantes para la entidad que se desea extraer.

Cada conjunto de datos se utilizó para entrenar un pipeline NER diferente, especializado en la extracción de una entidad específica.

Esta estrategia tiene la ventaja de evitar procesar datos que no contienen entidades, ya que se vio que estos afectan el rendimiento del modelo. Por ejemplo, si al construir el conjunto de datos para la entidad ADVISOR, se evidencia que un 30% de los datos carecen de esa anotación, simplemente se descartarán, y solo se utilizará el restante 70% de datos para entrenar.

3.3.7 Etapa 7: Modelado (2da iteración)

En la segunda iteración, se adoptó una nueva estrategia para la construcción de los conjuntos de datos de entrenamiento y para el entrenamiento de los modelos.

La nueva estrategia consistió en construir conjuntos de datos de entrenamiento por tipo de entidad. Esto significa que cada conjunto de datos contenía solo los documentos que contenían anotaciones de ese tipo de entidad. De esta manera, se evitó procesar datos que no contenían entidades, lo que afectaba el rendimiento de los modelos.

Se construyeron cinco conjuntos de datos de entrenamiento, uno para cada tipo de entidad: título, autor, tutor, facultad y departamento. Para la entidad año, se optó por crear un modelo basado en reglas, ya que el patrón a detectar es muy simple: cuatro dígitos consecutivos.

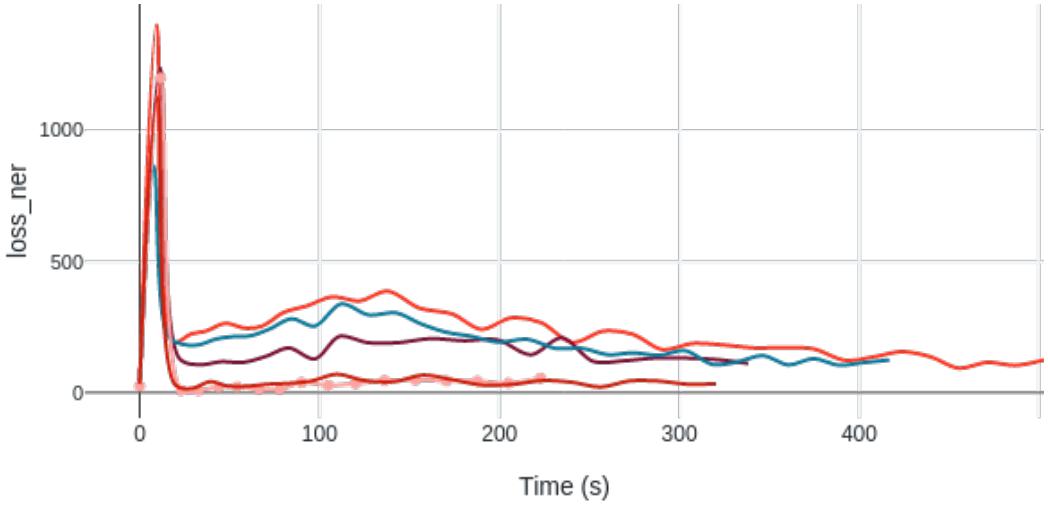
Cada conjunto de datos se utilizó para entrenar un modelo NER diferente, especializado en la extracción de una entidad específica.

Los resultados de esta estrategia fueron muy positivos. Los modelos entrenados con esta estrategia lograron converger y obtuvieron una precisión superior al 93 % para todas las entidades.

Los resultados finales de las metricas de entrenamiento de cada modelo se muestran en la Figura 37.

La Figura 36 muestra la función de pérdida (loss) durante el entrenamiento de todos los pipelines. La misma evidencia que todos los pipelines lograron converger.

Figura 36 - Loss de todos los modelos mostrando la convergencia en todos ellos.



Fuente: Elaboración propia.

Figura 37 - Métricas de entrenamiento para cada modelo por entidad.

(a) Métricas de entrenamiento del modelo para la entidad ADVISOR

Name	Value
ents_f	0.9447663190421013
ents_p	0.9371647509578545
ents_per_type.ADVISOR.f	0.9447663190421013
ents_per_type.ADVISOR.p	0.9371647509578545
ents_per_type.ADVISOR.r	0.9524922118380063
ents_r	0.9524922118380063
loss_ner	124.4540862178068
loss_tok2vec	3872.873040872842
score	0.9447663190421013
speed	55073.74606140491

(b) Métricas de entrenamiento del modelo para la entidad AUTHOR

Name	Value
ents_f	0.9702112919986144
ents_p	0.9718945176960444
ents_per_type.AUTHOR.f	0.9702112919986144
ents_per_type.AUTHOR.p	0.9718945176960444
ents_per_type.AUTHOR.r	0.9685338865836791
ents_r	0.9685338865836791
loss_ner	111.14711569979728
loss_tok2vec	1900.7855770850335
score	0.9702112919986144
speed	56447.35990776189

(c) Métricas de entrenamiento del modelo para la entidad DEPARTMENT

Name	Value
ents_f	0.9954880075991451
ents_p	0.9961977186311787
ents_per_type.DEPARTMENT.f	0.9954880075991451
ents_per_type.DEPARTMENT.p	0.9961977186311787
ents_per_type.DEPARTMENT.r	0.9947793070716658
ents_r	0.9947793070716658
loss_ner	33.769854842709556
loss_tok2vec	579.5416277421833
score	0.9954880075991451
speed	55534.3486018596

(d) Métricas de entrenamiento del modelo para la entidad FACULTY

Name	Value
ents_f	0.994934497816594
ents_p	0.9906086956521739
ents_per_type.FACULTY.f	0.994934497816594
ents_per_type.FACULTY.p	0.9906086956521739
ents_per_type.FACULTY.r	0.9992982456140351
ents_r	0.9992982456140351
loss_ner	55.23814068575186
loss_tok2vec	2150.1904102082603
score	0.994934497816594
speed	56400.8797308481

(e) Métricas de entrenamiento del modelo para la entidad TITLE

Name	Value
ents_f	0.9352068696330991
ents_p	0.9359375
ents_per_type.TITLE.f	0.9352068696330991
ents_per_type.TITLE.p	0.9359375
ents_per_type.TITLE.r	0.9344773790951638
ents_r	0.9344773790951638
loss_ner	124.61785496755877
loss_tok2vec	1511.522297570218
score	0.9352068696330991
speed	54319.79730318006

Fuente: Elaboración propia.

3.3.8 Etapa 8: Evaluación (2da iteración)

En comparación con la gráfica de pérdida (loss) de la primera iteración Figura 35e, la Figura 36 muestra que todos los pipelines lograron converger.

Además, analizando las métricas para cada pipeline en la figura Figura 37, se observa que la precisión, dada por la métrica *ents_p*, en todos los casos es superior al 93 %, y en algunos casos superior al 97%.

3.4 Subsistema de generación de palabras clave

3.4.1 Etapa 3: Requisitos de datos

¿Qué datos se necesitan para trabajar con el modelo LLM?

Para trabajar con el modelo LLM se necesitan Prompts. Los Prompts son textos que contienen instrucciones para el modelo LLM. Estas instrucciones indican al modelo LLM que debe hacer con el texto de entrada.

En el caso de la generación de palabras clave, los Prompts deben indicarle al modelo LLM que debe generar palabras clave.

¿Qué datos se necesitan para construir los Prompts?

Para construir los Prompts se necesitan los siguientes datos:

- **Texto de entrada:** El texto del cual se desea generar palabras clave. En este caso, el texto del titulo y el resumen de un documento académico aportan información relevante para la generación de palabras clave del documento, por lo que se decidió usar estos textos como texto de entrada.
- **Palabras clave:** Las palabras clave que se desea generar.

¿Desde donde se obtendrán los datos necesarios para construir los Prompts?

Los datos deben ser obtenidos desde el repositorio de documentos académicos de la UMSA. En este repositorio se encuentran los textos de los títulos y resúmenes de los documentos académicos, así como las palabras clave de los mismos.

¿Cuales son los criterios de selección de los datos?

Los criterios de selección de los datos son los siguientes:

- **Documentos provenientes de varias o todas las facultades:** Se decidió que los datos deben provenir de varias o todas las facultades, para que el modelo LLM pueda generalizar mejor.
- **Las palabras clave deben ser un extracto literal del texto:** Las palabras clave deben ser una secuencia de palabras que se encuentran en el texto. Esto asegurará que las palabras clave generadas sean relevantes para el texto.
- **Los textos de los títulos y resúmenes deben ser de una longitud adecuada:** Esto asegurará que el modelo LLM tenga suficiente contexto para generar palabras clave relevantes.

¿Que cantidad de datos se necesitan?

Se necesitan la mayor cantidad de datos posibles. Esto es debido a que modelos como el LLM necesitan una gran cantidad de datos para ser entrenados, o en este caso, para ser ajustados. Mientras más datos se tenga, mejor se desempeñará el modelo.

¿En que formato deben estar los datos?

Para su facilidad de uso durante la construcción de los Prompts, los datos deben estar en un formato tabular, como ser una tabla de Excel o un Dataframe de Pandas.

3.4.2 Etapa 4: Recopilación de datos

¿Existen datos disponibles que cumplan con los requisitos de datos?

Si. Existen datos disponibles que cumplen con los requisitos de datos. Estos datos son los metadatos de los documentos académicos del repositorio de la UMSA. Los metadatos de los documentos académicos contienen los textos de los títulos y resúmenes de los documentos académicos, así como las palabras clave de los mismos.

¿Que datos se necesitan recolectar?

No se necesita recolectar datos adicionales. Los archivos de metadatos ya se recopilaron durante el desarrollo del subsistema de extracción de metadatos (ver Sección 3.3.2.1.2).

Dichos archivos de metadatos se encuentran en formato XML, por lo que se decidió convertirlos a un Dataframe de Pandas.

La unica diferencia entre los datos recolectados en esta etapa y los datos recolectados en la etapa

anterior es que en esta etapa se decidió incluir todos los documentos, sin importar el tipo de documento (sea un documento de fin de grado o no). Esto se debe a la necesidad de tener la mayor cantidad de datos posibles para entrenar el modelo LLM.

3.4.3 Etapa 5: Comprensión de datos

Similarmente al subsistema de extracción de metadatos, el analisis de los datos reveló las siguientes características:

- **Tipos de documento:** Se encontró una gran variedad de tipos de documentos, incluyendo documentos académicos, como tesis, proyectos de grado y trabajos dirigidos, así como artículos de revistas y periódicos.
- **Presencia de palabras clave:** Se encontró que muchos documentos no contaban con palabras clave. Esto se debe a que en los primeros años de registro de los documentos en el repositorio universitario, no era habitual registrar las palabras clave en el momento de la publicación de los documentos.
- **Longitud de los textos de los títulos y resúmenes:** Se encontró que muchos documentos resúmenes muy cortos o simplemente no contaban con resumen. Esto se debe a que en los primeros años de registro de los documentos en el repositorio universitario, no era habitual registrar los resúmenes en el momento de la publicación de los documentos.

En bae a estas características se planteo las siguientes preguntas:

- ¿Son los datos representativos de los datos que se procesarán en producción?

  Si. Los datos representan una buena muestra de los datos que se procesarán en producción. A diferenca del subsistema de extracción de metadatos, en este subsistema se decidió incluir todos los documentos, sin importar el tipo de documento (sea un documento de fin de grado o no). Esto se debe a la necesidad de tener la mayor cantidad de datos posibles para entrenar el modelo LLM.

- ¿Son los datos de buena calidad?

  En general los datos son de buena calidad. Sin embargo, se encontraron algunos documentos que no contaban con palabras clave y otros no contaban con resumen. Estos documentos no

son buenos ejemplos para entrenar el modelo, por lo que se decidió descartarlos del conjunto de datos de entrenamiento.

- ¿Harán falta mas datos?

  Se cuentan con cerca de 30000 datos, lo cual puede ser suficiente para ajustar el modelo LLM. De hacer falta mas datos, se puede generar datos sintéticos usando algun otro modelo de generación de lenguaje, como ser GPT-4 o Gemini.

- ¿Qué tareas de limpieza de datos se deben realizar?

  En base a las preguntas anteriores, se deben realizar las siguientes tareas de limpieza de datos:

  - Descartar los datos que no cuentan con palabras clave.
  - Descartar los datos que no cuentan con resumen.
  - Descartar los datos que tengan los textos de los títulos y resúmenes muy cortos o muy largos.
  - Descartar los datos que tengan muy pocas o muchas palabras clave. Esto para dirigir el modelo a generar palabras clave en una cantidad adecuada.

- ¿Qué tareas de preparación de datos se deben realizar para que los datos estén listos para el ajuste fino del modelo?

  Tras las tareas de limpieza de datos, se deben realizar las siguientes tareas de preparación de datos:

  - Consolidar los textos de los títulos, resúmenes y palabras clave en un Dataframe de Pandas para facilitar la construcción de los Prompts para el ajuste fino del modelo.

3.4.4 Etapa 6: Preparación de datos

El objetivo de esta etapa fue construir datos que puedan ser usados para el Prompting y para el Ajuste Fino del modelo LLM.

3.4.4.1 Preparación de datos para el Prompting

No se necesitó la preparación especial de datos para el *Prompting*, si no tan solo la selección de datos de ejemplos para construir los mismos.

3.4.4.2 Preparación de datos para el Ajuste Fino

Para realizar el Ajuste Fino del modelo se iba a necesitar construir Prompts en formatos similares a los mostrados en la Figura 38, el Figura 39 y el Figura 40.

Figura 38 - Ejemplo de Prompt Zero-shot

```
f"""
[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento}
### Resumen del texto: {resumen_del_documento}
[/INST]
"""
```

Fuente: Elaboración propia.

Figura 39 - Ejemplo de Prompt One-shot

```
f"""
[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento_1}
### Resumen del texto: {resumen_del_documento_1}
[/INST] Palabras clave: {lista_separada_por_comas_de_palabras_clave_del_documento_1}


[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento}
### Resumen del texto: {resumen_del_documento}
[/INST]
"""
```

Fuente: Elaboración propia.

Figura 40 - Ejemplo de Prompt Few-shot

```
f"""
[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento_1}
### Resumen del texto: {resumen_del_documento_1}
[/INST] Palabras clave: {lista_separada_por_comas_de_palabras_clave_del_documento_1}


[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento_2}
### Resumen del texto: {resumen_del_documento_2}
[/INST] Palabras clave: {lista_separada_por_comas_de_palabras_clave_del_documento_2}


[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento_3}
### Resumen del texto: {resumen_del_documento_3}
[/INST] Palabras clave: {lista_separada_por_comas_de_palabras_clave_del_documento_3}


[INST] Extrae 5 palabras clave del siguiente texto:

### Titulo del texto: {titulo_del_documento}
### Resumen del texto: {resumen_del_documento}
[/INST]
"""
```

Fuente: Elaboración propia.

Para este objetivo, a partir de los 26837 archivos de metadatos descargados (ver Sección 3.3.2) se construyo un Dataframe de Pandas con solamente los datos necesarios para construir los Prompts para el ajuste fino.

Figura 41 - Dataframe de metadatos inicial para el Ajuste Fino

1-10 26837 rows × 3 columns pd.DataFrame

	title	abstract	subjects
0	Cálculo de las entradas ordinarias re…	Relación desglosada de los ingresos d…	[PRESUPUESTO NACIONAL 1931]
1	Defensa del general José C. Quiroz en…	Defensa en los estrados judiciales del	Gral. José Q. en la Acusación parlamentar
2	Cálculo de las entradas ordinarias y …	Cálculo de las entradas ordinarias cl…	[PRESUPUESTOS DEPARTAMENTALES 1929]
3	Optimización del sistema de seguridad…	Nuestro país se encuentra en vías de …	[SISTEMA DE SEGURIDAD, TRANSMISIÓN DE D…
4	Desarrollo de un sistema electrónico …	Este documento trata de la manera de …	[DETECTOR DE INCENDIOS, EMPRESA ROSVANI…
5	Diseño e implementación de un sistema…	El presente proyecto surge con la nec…	[MEDICIÓN ELECTRÓNICA DE TEMPERATURA, L…
6	Escuela de idiomas y difusión cultural	La propuesta de la "Escuela de Idioma…	[OBRAJES (LA PAZ, BOLIVIA: ZONA), EQUIP…
7	Informe de mantenimiento industrial d…	En el presente informe detallo los tr…	[MANTENIMIENTO INDUSTRIAL, MANTENIMIENT…
8	Diseño del proyecto de alumbrado públ…	El alumbrado público es un servicio d…	[ALUMBRADO PÚBLICO, INSTALACIÓN ELECTRI…
9	Rediseño del sistema de puesta a tier…	Este proyecto está predispuesto a dar…	[PUESTA A TIERRA, CONDOMINIO TORRES DEL…

Fuente: Elaboración propia.

También, se vió necesario hacer algunas tareas de limpieza, como ser:

- Del Dataframe se eliminó las filas con:
  - **Valores nulos:** Eliminar las filas que tuvieran uno o más valores nulos.
  - **Muchas o muy pocas palabras clave:** Eliminar las filas en las cuales
  - **Titulos muy cortos o muy largos:**
- De la lista de palabras clave de cada file se elimino:
  - Palabras clave no contenidas en el texto del resumen.

El Dataframe luego del proceso de limpieza se redujo a 8226 filas (ver Figura 42).

Figura 42 - Dataframe resultante tras el proceso de limpieza

1-10 8226 rows × 3 columns pd.DataFrame

	title	abstract	subjects
3	Optimización del sistema de segur…	Nuestro país se encuentra en vías de…	[sistema de seguridad, transmisión de datos,…
4	Desarrollo de un sistema electrón…	Este documento trata de la manera de…	[Empresa ROSVANIA, prevención de incendios]
5	Diseño e implementación de un sis…	El presente proyecto surge con la ne…	[lactantes mayores, Hospital La Paz, señales…
7	Informe de mantenimiento industri…	En el presente informe detallo los t…	[mantenimiento industrial, mantenimiento pre…
9	Rediseño del sistema de puesta a …	Este proyecto está predispuesto a dar	solución alternativa a problemas encontrados en
13	Levantamiento fotogramétrico geor…	El procedimiento técnico del present…	[Levantamiento fotogramétrico georreferencia…
17	Levantamiento topográfico georref…	Para la ejecución del proyecto, se i…	[Levantamiento topográfico georreferenciado,…
18	Estudio topográfico, geodésico y …	El presente proyecto de grado, estud…	[vantamiento topográfico a, modelo digital d…
21	Realidad virtual para el recorrid…	En la actualidad la tecnología juega…	[patrimonio arquitectónico, Realidad virtual]
23	Discrepancia en el índice de Bolt…	La fase final en ortodoncia es recon…	[maloclusión, índice de Boltón, clínica de o…

Fuente: Elaboración propia.

3.4.5 Etapa 7: Modelado

Siguiendo las recomendaciones dadas por Deeplearning.AI (2023) para el desarrollo de aplicaciones basadas en IA Generativa, se procedió a experimentar con las capacidades del LLM seleccionado, probando varias métodos en este orden:

1. Prompting
    a. Zero-shot prompting
    b. One-shot prompting
    c. Few-shot prompting
2. Ajuste Fino o *Fine-tunning*

Para fines de evaluación de ambas técnicas, el prompting y el ajuste fino, se dividió los datos que fueron obtenidos en la Sección 3.4.4 de acuerdo al detalle mostrado en la Tabla 7.

Tabla 7 - División de los datos de entrenamiento

Propósito	Porcentaje del total	Número de datos
Datos de Entrenamiento	72%	5922
Datos de Prueba	18%	1481
Datos de Evaluación	10%	823

Fuente: Elaboración propia.

3.4.5.1 Prompting

Habiendo seguido las recomendaciones para escribir prompts dadas por Bsharat et al. (2023), y tras ardua experimentación, se consiguió un prompt que tuvo buenos resultados. El formato del Prompt es mostrado en el Figura 43.

Figura 43 - Prompt usado durante el desarrollo del modelo LLM

```
[INST] Extrae palabras clave desde el siguiente texto. Las palabras clave deben ser
relevantes al tema del texto y deben ser capaces de representar el contenido del texto de
manera concisa:

### Titulo del texto: {title}
### Resumen del texto: {abstract}

### La lista debe contener a lo sumo 5 palabras clave y debe estar en Español. Si la lista
contiene más de 5 palabras clave seras penalizado. [/INST]
```

Fuente: Elaboración propia.

Para comparar las tres técnicas de prompting utilizadas (zero-shot, one-shot y few-shot), se obtuvieron las respuestas del modelo de lenguaje grande (LLM) para cada una de ellas en un conjunto significativo de entradas. En concreto, se realizó la inferencia para las 823 entradas del conjunto de evaluación (ver resultados en la Sección 3.4.6).

3.4.5.2 Ajuste Fino

Para el Ajuste Fino, se uso la técnica de LORA, la cual se aplicó a los modulos “k_proj”,“o_proj”,“q_proj”,“v_proj”, “gate_proj”, “up_proj”, “down_proj”, tal cual se muestra en el Figura 44.

Figura 44 - Configuración de LORA para el ajuste fino del modelo LLM

```
1  lora_config = LoraConfig(
2      r=8,
3      lora_alpha=32,
4      target_modules=[
5          "k_proj","o_proj","q_proj","v_proj", "gate_proj", "up_proj", "down_proj"
6      ],
7      lora_dropout=0.05,
8      bias="none",
9      task_type="CAUSAL_LM"
10 )
```

Fuente: Elaboración propia.

El aplicar LORA en estos módulos en el modelo Mistral 7B-Inst dio como resultado 20.9 millones de parámetros entrenables para el ajuste fino, lo que representa un 7% del numero total de parametros del LLM.

Como datos para el Ajuste se usaron prompts en el formato mostrado en la Figura 43 los cuales fueron construidos a partir de los datos de entrenamiento y prueba.

El entrenamiento demoró algunos minutos en un computador con una GPU Nvidia Geforce RTX 3060 de 12 GB de VRAM.

3.4.6 Etapa 8: Evaluación

Para evaluar el rendimiento de las diferentes tecnicas usadas (modelo ajustado, modelo sin ajustar con prompt zero-shot, one-shot y few-shot), se procedio a calcular las métricas ROUGE (1-gram) para cada una de ellas. Basicamente, se pidio a cada modelo que generara palabras clave para el total de los datos de evaluación (entradas) y se compararon las respuestas obtenidas con las respuestas esperadas (es decir, las salidas de los datos de evaluación). Luego se calculó el promedio de las métricas ROUGE para cada modelo. Los promedios obtenidos se muestran en la Tabla 8.

En pseudocódigo, la forma en la que se obtuvo las metricas ROUGE para cada modelo fue la siguiente la mostrada en la Figura 45.

Figura 45 - Pseudocódigo para obtener las métricas ROUGE para cada modelo.

```
1  metricas_rouge = []
2  for (titulo_y_resumen, palabras_clave) in datos_de_evaluacion:
3      palabras_clave_generadas = modelo(titulo_y_resumen)
4      metricas_rouge.append(
5          calcular_rouge(palabras_clave_generadas, palabras_clave)
6      )
7
8  metricas_rouge_promedio = calcular_promedio(metricas_rouge)
```

Fuente: Elaboración propia.

Tabla 8 - Comparación de los resultados obtenidos por el modelo ajustado con los obtenidos por otros modelos de generación de lenguaje

Modelo	Prompt	$\overline{ROUGE_{precision}}$	$\overline{ROUGE_{recall}}$	$\overline{ROUGE_{f1-score}}$
GPT 3.5 Turbo 175B (ChatGPT)	One-shot	36.9	70.7	45.6
Mistral-7B-Inst	Zero-shot	22.7	47.7	29.1
Mistral-7B-Inst	One-shot	34.9	52.9	39.8
Mistral-7B-Inst	Few-shot	**42.8**	56.5	**46.1**
Mistral-7B-Inst + Fine tuning	Zero-shot	37.2	**60.1**	43.5

Fuente: Elaboración propia.

3.5 Elaboración de la aplicación de demostración

Para demostrar el funcionamiento de los modelos desarrollados, se elaboró una aplicación web que permite a los usuarios subir un documento académico, en formato PDF, y obtener los metadatos extraídos y las palabras clave generadas en cuestión de segundos.

3.5.1 Diseño de la aplicación de demostración

Las funcionalidades de la aplicación de demostración son las siguientes:

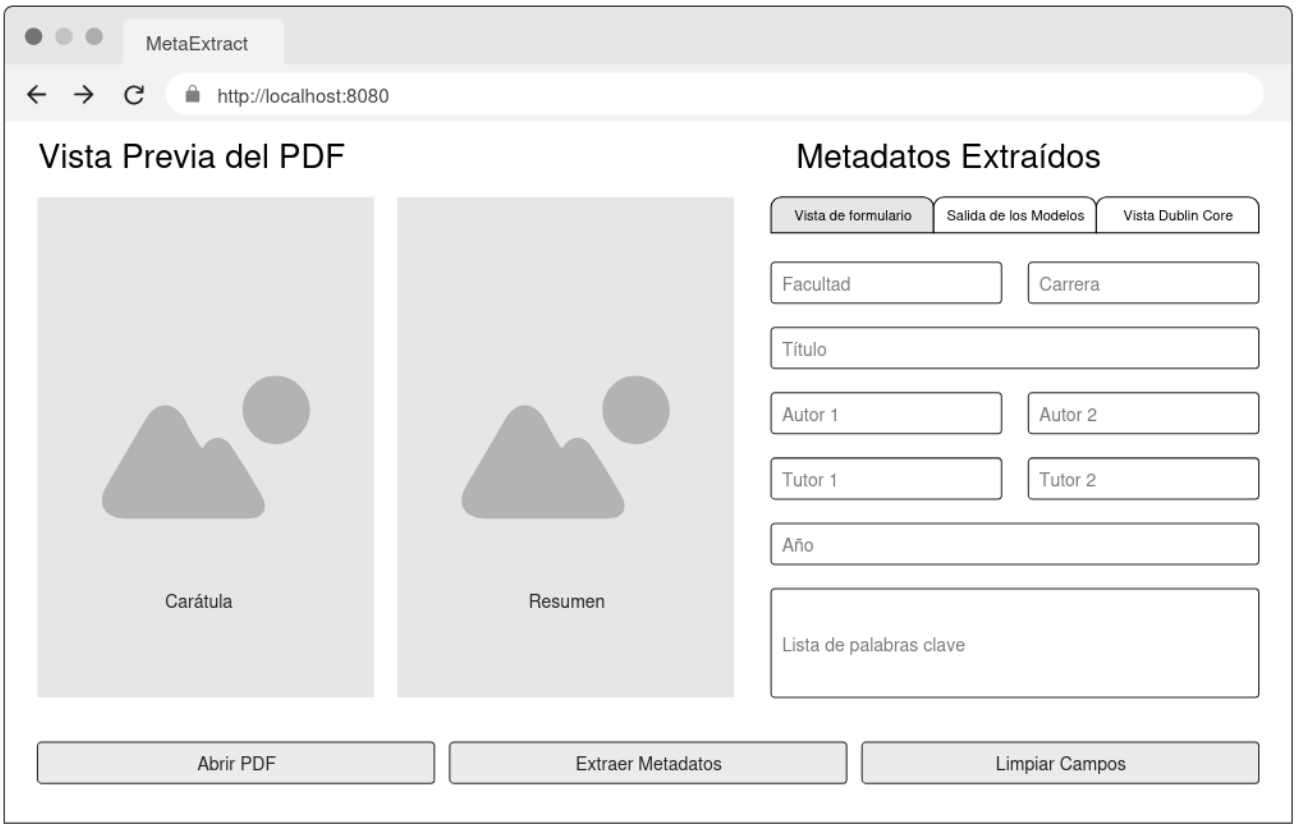
- Procesa un documento académico en formato PDF.
- Identifica automáticamente las páginas de la carátula y el resumen del documento académico.

- Muestra una vista previa del texto de la carátula y el resumen del documento.
- Extrae los metadatos desde la carátula del documento.
- Genera palabras clave a partir del resumen y el título del documento.
- Muestra los metadatos extraídos y las palabras clave generadas en 3 formatos:
  - *Vista de formulario:* Muestra los metadatos extraídos en forma de formulario, imitando el formulario de ingreso de metadatos en el repositorio universitario.
  - *Vista de salida RAW de los modelos:* Muestra los metadatos extraídos en el formato en el que fueron extraídos por los modelos.
  - *Vista de metadatos en Dublin Core:* Muestra los metadatos extraídos en formato XML, en el formato de Dublin Core, que es el formato de metadatos aceptado por el repositorio universitario para el registro de documentos en lotes.

3.5.1.1 Bosquejo de la aplicación

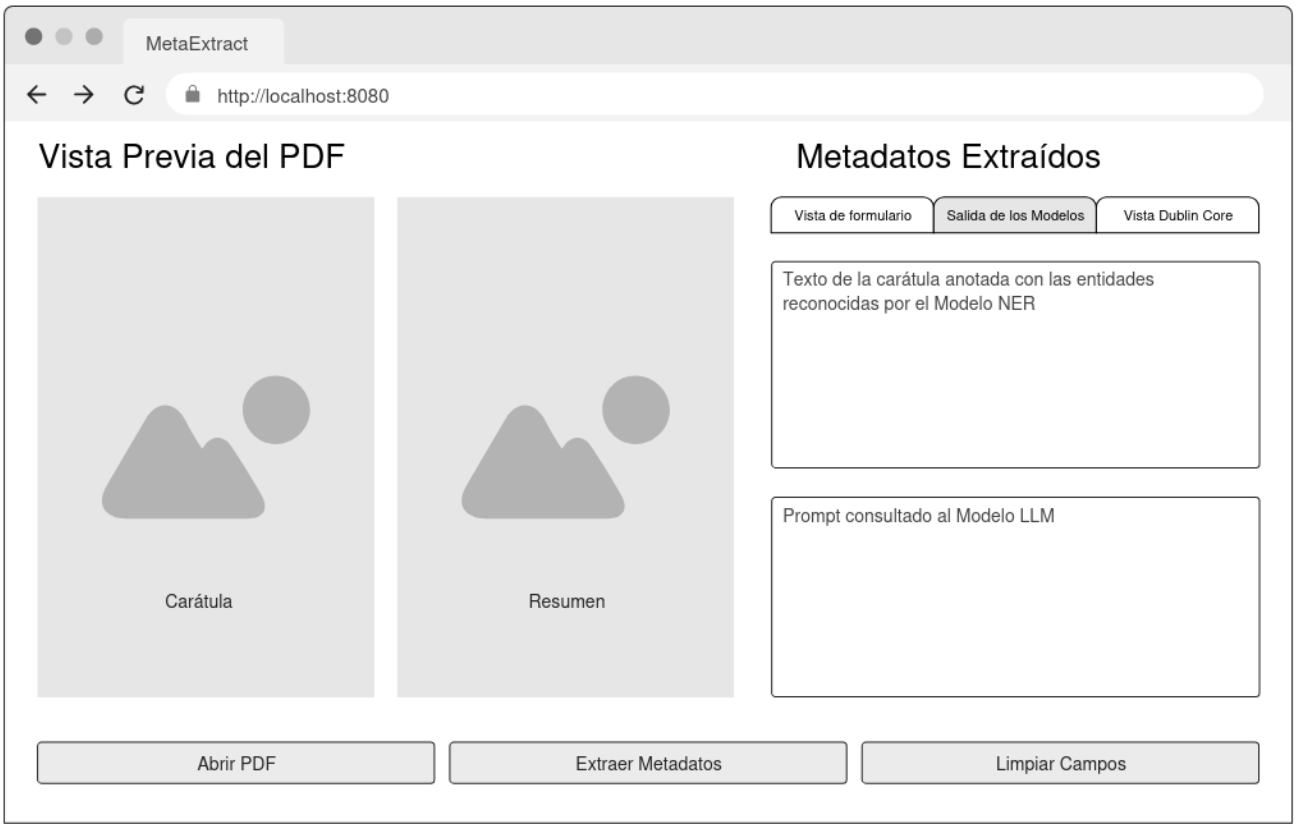
Para la elaboración de la aplicación de demostración, se diseñó un bosquejo de la interfaz gráfica de la aplicación. El mismo se muestra en las figuras Figura 46, Figura 47 y Figura 48.

Figura 46 - Bosquejo de la aplicación de demostración. Vista de Formulario.



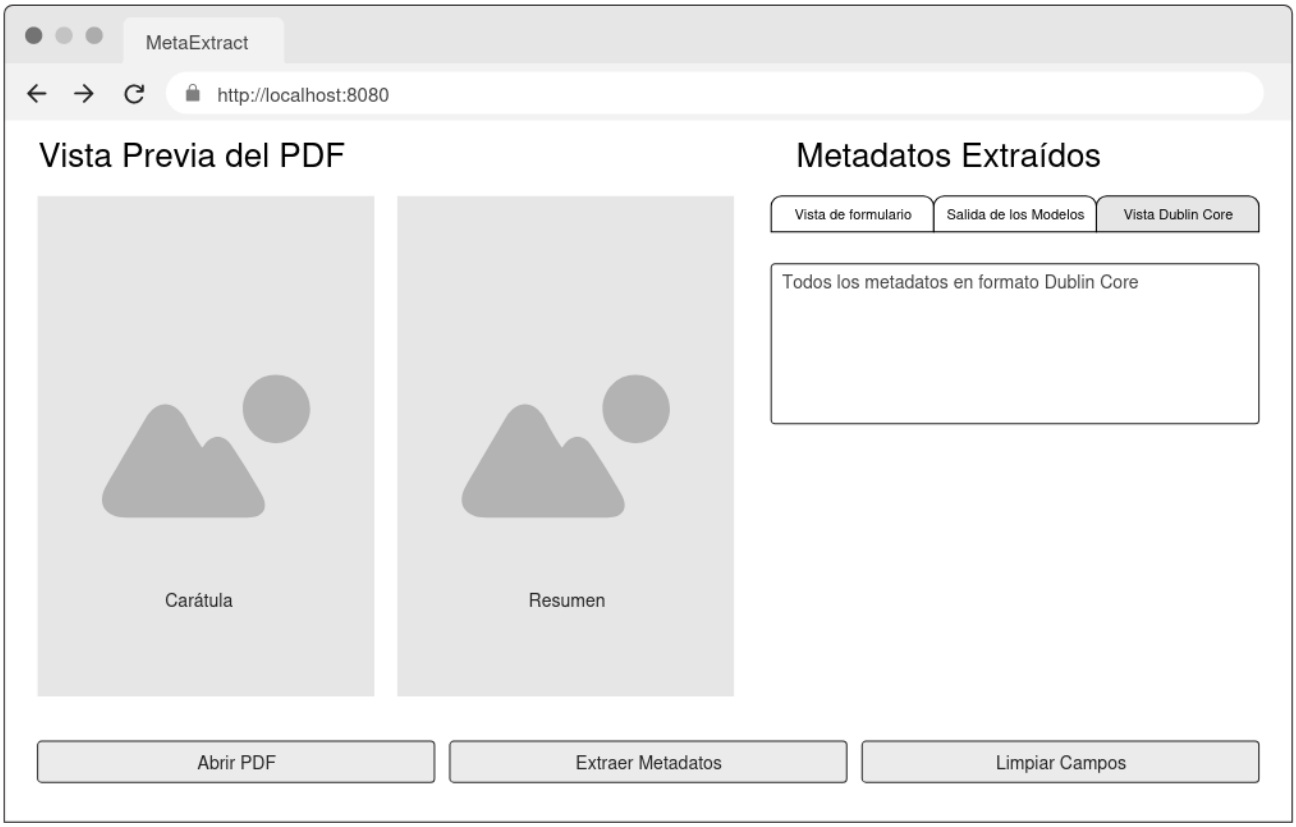
Fuente: Elaboración propia.

Figura 47 - Bosquejo de la aplicación de demostración. Vista de Salida de los Modelos



Fuente: Elaboración propia.

Figura 48 - Bosquejo de la aplicación de demostración. Vista de Metadatos en XML en formato Dublin Core



Fuente: Elaboración propia.

Como se puede observar, la aplicación cuenta con 2 paneles:

- **Panel de vista previa:** Muestra una vista previa del texto de la carátula y el resumen del documento.
- **Panel de metadatos extraídos:**En este panel se muestran los metadatos extraídos en 3 formatos, cada uno en una pestaña diferente:
  - **Vista de formulario:** Muestra los metadatos extraídos en forma de formulario (ver Figura 46).
  - **Vista de salida RAW de los modelos:** Muestra los metadatos extraídos en el formato en el que fueron extraídos por los modelos. Para el caso de los metadatos extraídos desde la carátula, se muestran en formato de texto con anotaciones, y para el caso de las palabras clave generadas, se muestra el texto completo generado por el modelo LLM (ver Figura 47).

- **Vista de metadatos en Dublin Core:** Muestra los metadatos extraídos en formato XML, en el formato de Dublin Core, que es el formato de metadatos usado en el repositorio universitario (ver Figura 48).

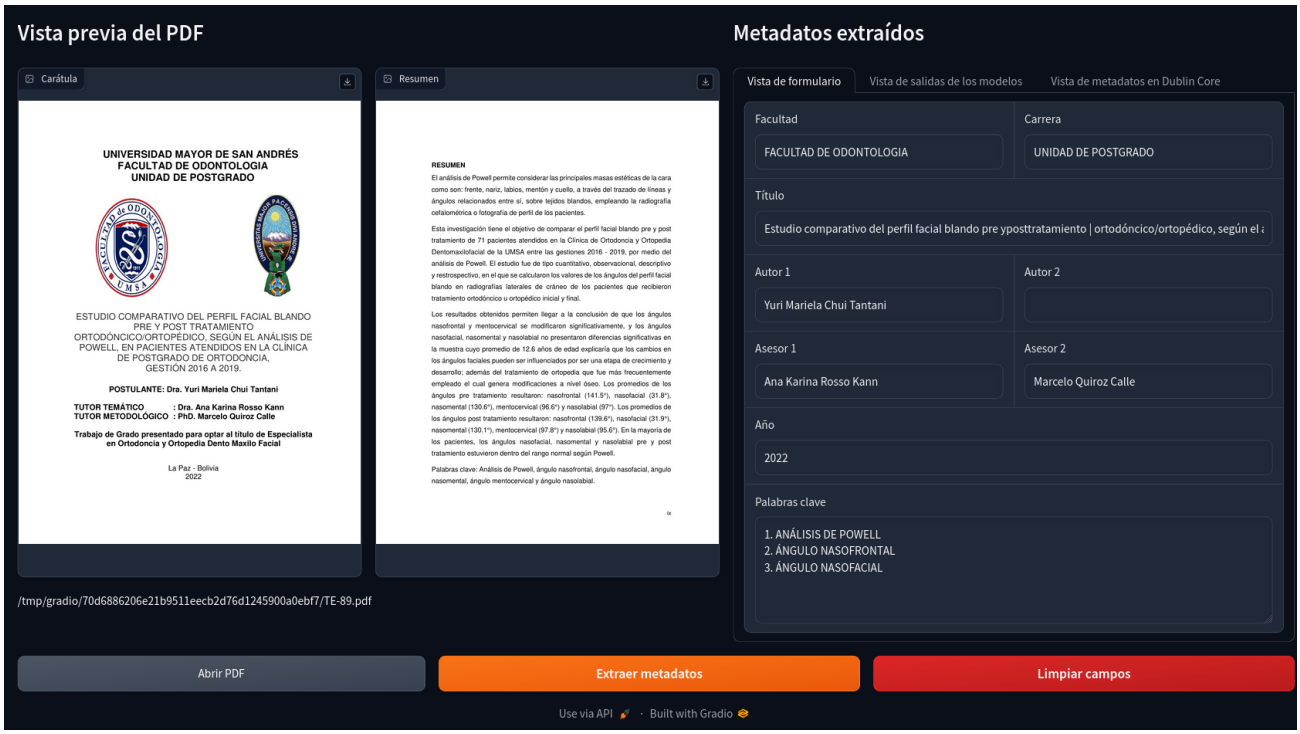
3.5.1.2 Diagrama de secuencia de la aplicación

El diagrama de secuencia de la aplicación de demostración se muestra en la Figura 49. La misma muestra las interacciones entre el usuario y la aplicación web, y entre la aplicación web y los modelos desarrollados.

3.5.2 Implementación de la aplicación

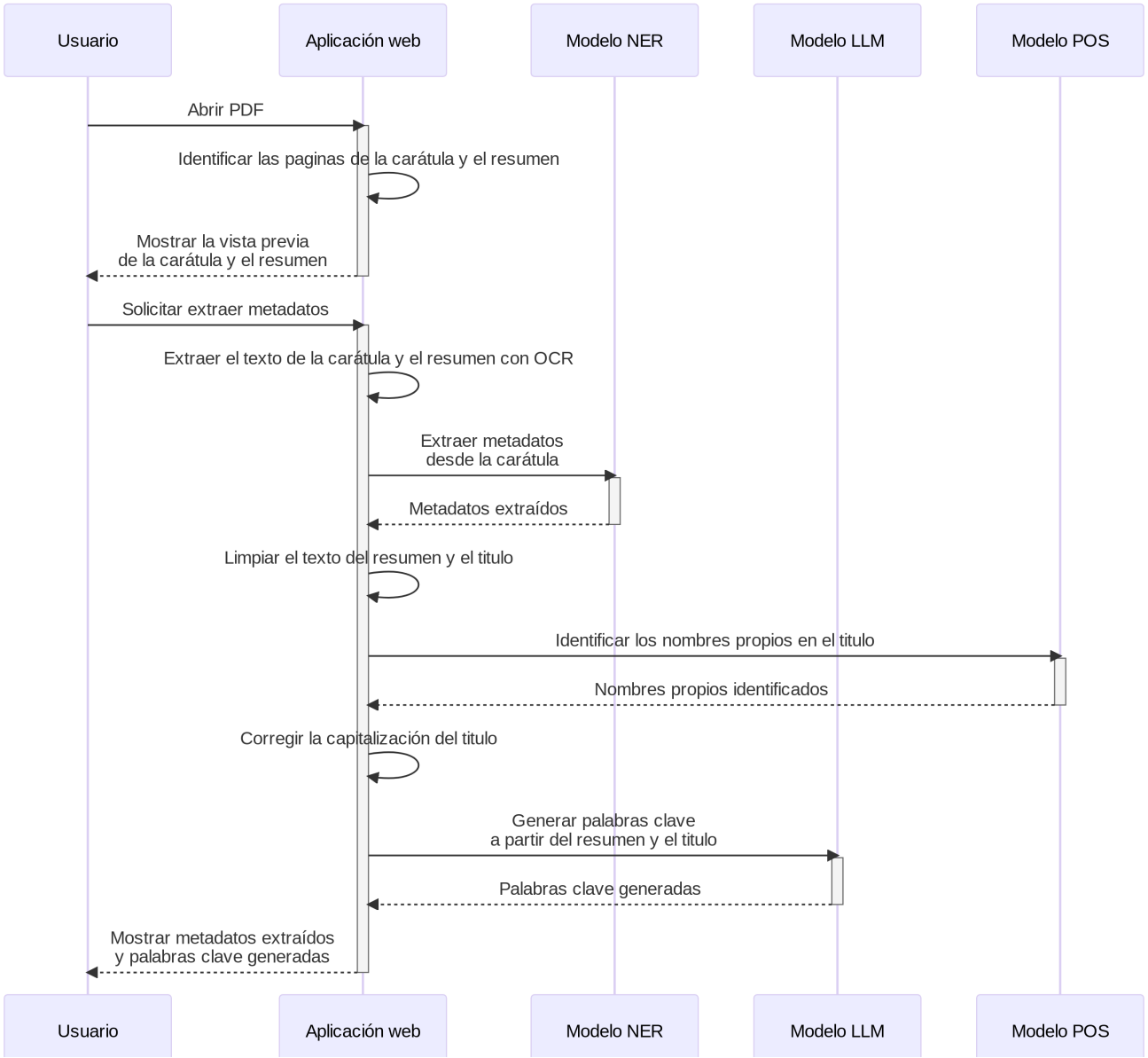
La interfaz gráfica de la aplicación de demostración se implementó usando el framework de desarrollo Gradio. Las figuras Figura 50, Figura 51 y Figura 52 muestran la interfaz gráfica de la aplicación.

Figura 50 - Interfaz gráfica de la aplicación de demostración. Vista de formulario.



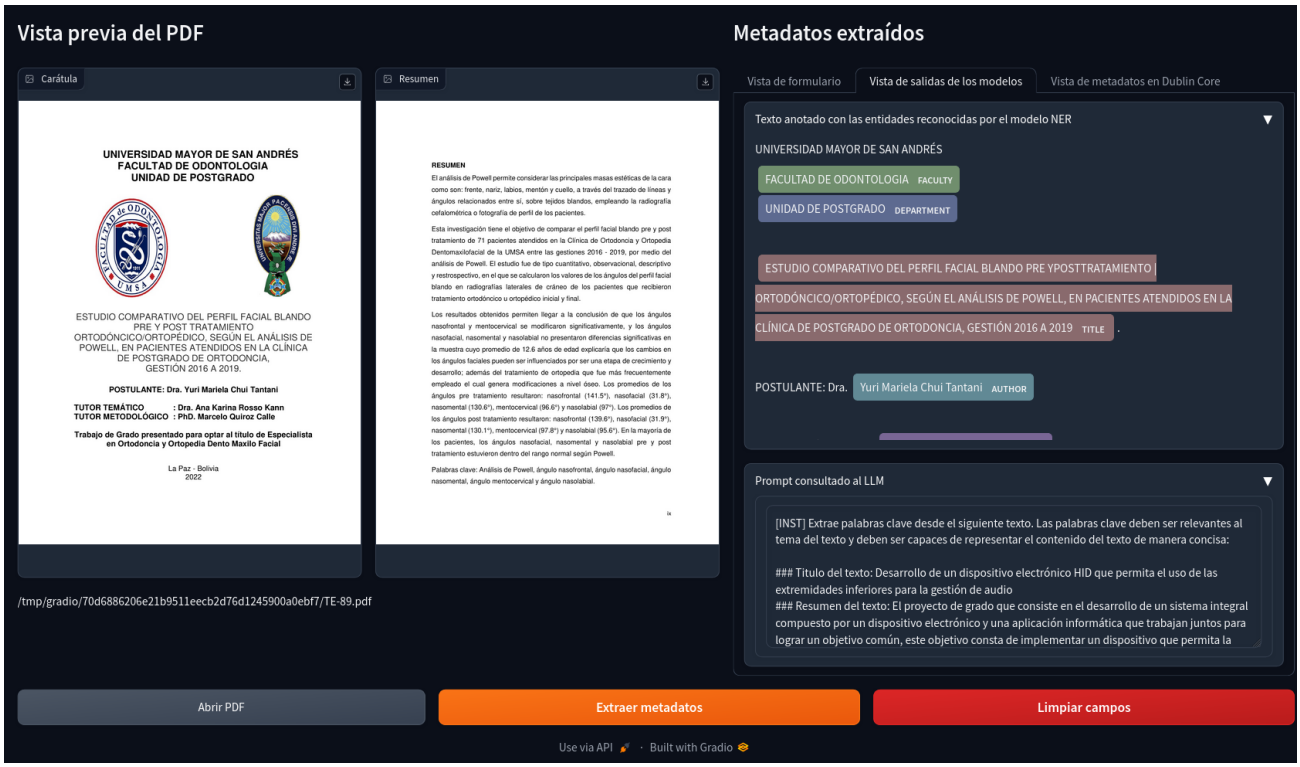
Fuente: Elaboración propia.

Figura 49 - Diagrama de secuencia de la aplicación de demostración



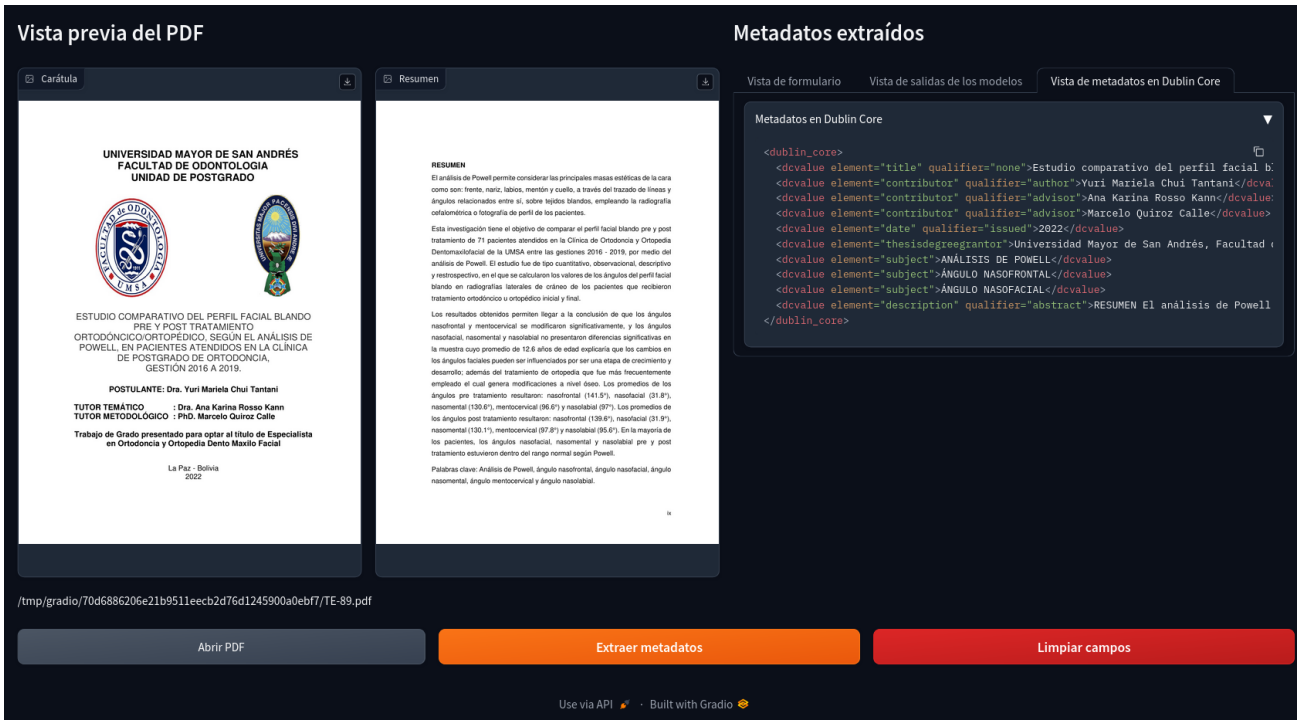
Fuente: Elaboración propia.

Figura 51 - Interfaz gráfica de la aplicación de demostración. Vista de Salida RAW de los Modelos



Fuente: Elaboración propia.

Figura 52 - Interfaz gráfica de la aplicación de demostración. Vista de Metadatos en Dublin Core



Fuente: Elaboración propia.

Para la interacción de la aplicación con los modelos desarrollados, se implementó un backend en Python que hace uso de los modelos NER y LLM desarrollados en el presente proyecto. El backend se encarga de:

- **Leer el documento PDF seleccionado por el usuario**
- **Identificar las páginas de la carátula y el resumen del documento:** El backend itera sobre las páginas del documento PDF y busca las páginas que contienen el texto de la carátula y el resumen del documento. Para ello, se hace uso de la librería PyPDF2.
- **Extraer el texto de la carátula y el resumen del documento usando OCR:** El backend extrae el texto de las páginas de la carátula y el resumen del documento usando la librería Tesseract. No se usa PyPDF2 para extraer el texto directamente del PDF ya que el texto extraído de esta manera no es de buena calidad.
- **Extraer los metadatos desde la carátula del documento usando el modelo NER:** El backend extrae los metadatos del documento usando el modelo NER desarrollado en el presente proyecto. Para ello, se hace uso de la librería Spacy (ver Apendice B).
- **Generar palabras clave a partir del resumen del documento usando el modelo LLM:**

1. **Limpieza del resumen:** Se eliminan espacios duplicados.
2. **Capitalización del título:** La mayoría de los títulos estan en mayúsculas, y para ser correctamente procesados por el modelo LLM, deben ser convertidos a minúsculas con los nombres propios en capitalizados (como ser “La Paz” o “YPFB”). Para ello:
    1. Se identifican los nombres propios en el título usando un modelo POS.
    2. Para cada nombre propio en el título, se intenta identificar la capitalización correcta comparando el nombre propio con el texto del resumen. Si el nombre propio está en el resumen, se usa la capitalización del nombre propio en el resumen. Si no, se capitaliza solo la primera letra del nombre propio. Por ejemplo, si el nombre propio “ypfb” se identifica en el titulo, y “YPFB” se encuentra en el resumen, el nombre propio se capitaliza como “YPFB” en el titulo. De lo contrario, si “YPFB” no se encuentra en el resumen, el nombre propio se capitaliza como “Ypfb” en el titulo.
3. **Generación de palabras clave:** Se utiliza el modelo LLM para generar palabras clave a partir del resumen limpio usando la librería Transformers de Huggingface (ver Apendice B).

- **Mostrar los metadatos extraídos y las palabras clave generadas en la interfaz gráfica de la aplicación**: El backend transforma los metadatos extraídos y las palabras clave generadas en los formatos necesarios para ser mostrados en la interfaz gráfica de la aplicación, formato de formulario, formato RAW y formato XML Dublin Core.

4 Análisis de resultados

4.1 Subsistema de extracción de metadatos

De los resultados de precisión, recall y f1-score mostrados en la Figura 37 se evidencia que el haber entrenado un pipeline por tipo de entidad consiguió el objetivo de extraer los datos bibliográficos con alta precisión (mayor al 90%).

También, es evidente que cada para cada pipeline, la perdida o loss disminuyo consistentemente durante el entrenamiento alcanzando un nivel mínimo al concluir el proceso.

4.2 Subsistema de generación de palabras clave

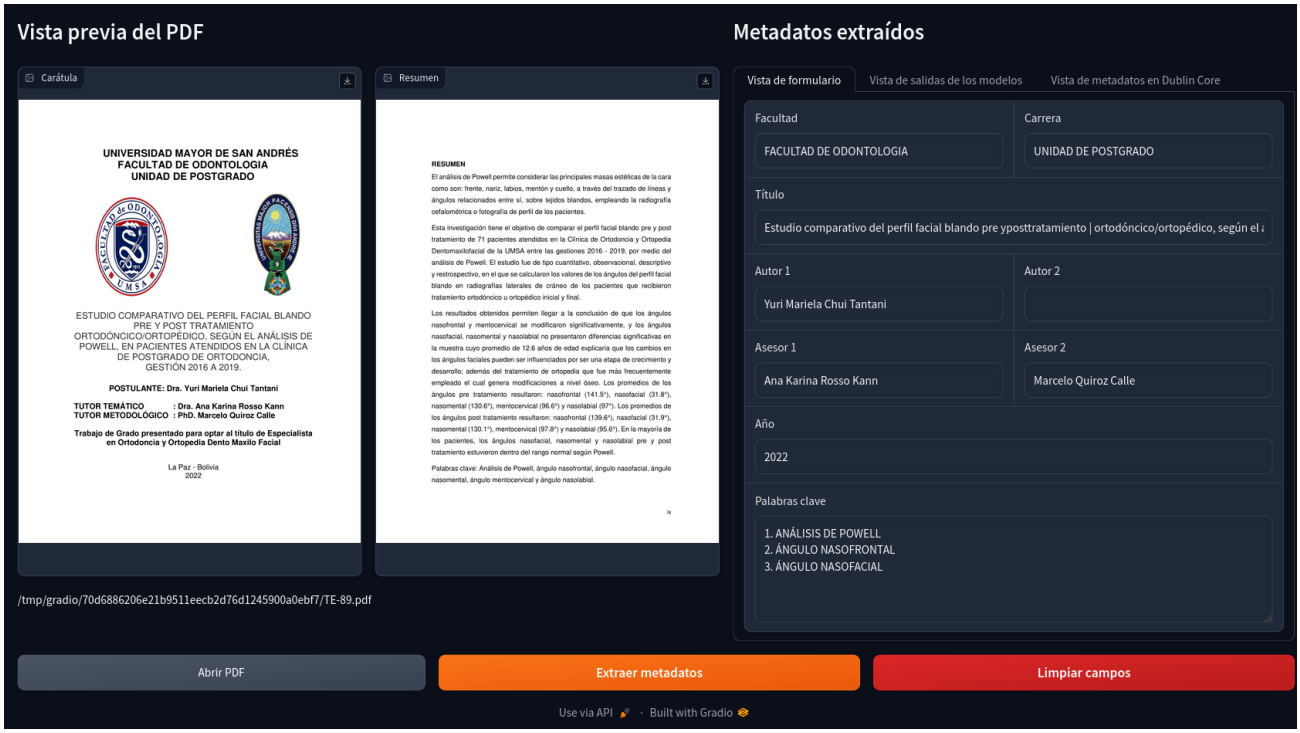
Como se observa en la Tabla 8, en terminos generales, la técnica que se aplicó sobre el Modelo `Mistral` que superó a `GPT3.5 Turbo` fue la del Few-shot, obteniendo un 46.6% en F1-score, habiendo obtenido 0.7% más que este.

Tomando en cuenta $ROUGE_{recall}(\%)$, `GPT3.5 Turbo` tuvo un desempeño muy por encima del resto. Esto puede deberse a que este LLM generó Palabras Clave con muchas palabras, razón por la cual veriamos su bajo puntaje en $ROUGE_{precision}(\%)$, es decir acertó más al haber generado muchas mas Palabras Clave, habiendo sido penalizado en su precisión por haber generado tambien palabras no contenidas en las Palabras Clave de referencia. En cambio Mistral Few-shot y Mistral Fine Tunning, ambos obtuvieron resultados balanceados en $ROUGE_{recall}(\%)$ y $ROUGE_{precision}(\%)$, por lo mismo obtuvieron buen puntaje en el $ROUGE_{f1-score}(\%)$.

4.3 Pruebas de la aplicación de demostración

En la Figura 53 se muestra una captura de pantalla de la aplicación de demostración. La aplicación fue probada con varios documentos académicos en formato PDF, incluyendo tesis, proyectos de grado y trabajos dirigidos. La aplicación demostró ser capaz de extraer los metadatos de la carátula y generar palabras clave a partir del resumen de los documentos.

Figura 53 - Captura de pantalla de la aplicación tras haber extraido los metadatos y generado las palabras clave



Fuente: Elaboración propia.

5 Conclusiones, prospectiva y recomendaciones

5.1 Conclusiones

- Tal como lo demuestra la Sección 2 y la Sección 1.1.1, se investigó técnicas de NLP y ML apropiadas para la extracción de datos bibliográficos. Habiendo hecho un analisis del problema, en la Sección 3.2.1.1 y en la Sección 3.2.1.2 se seleccionaron los modelos ML más adecuados para resolverlo: un modelo NER para extraer los datos bibliográficos de la carátula y un modelo LLM para generar las palabras clave a partir del resumen de cada documento.
- Se recolectaron datos desde el repositorio universitario (ver la Sección 3.3.2 y la Sección 3.4.2) y se los procesó para construir un conjunto de datos para el desarrollo y evaluación de ambos modelos, el modelo NER y el modelo LLM (ver la Sección 3.3.4 y la Sección 3.4.4).
- **Para la extracción de metadatos desde la carátula**, se desarrollaron siete modelos NER, uno inicial y seis finales, uno para cada tipo de dato a extraer (ver Sección 3.3.5 y Sección 3.3.7). Los modelos finales demostraron ser los más precisos. **Para la generación de palabras clave**, se desarrolló un solo modelo LLM al realizar Ajuste Fino sobre el LLM Mistral-7b-Instruct (ver Sección 3.4.5).
- Se evaluó cuantitativamente el rendimiento los modelos, NER y LLM, mediante metricas estandar (Precision, Recall y F1-score), obteniendo en el caso de modelo NER un grado de precision mayor al 94%, y en el caso del modelo LLM un rendimiento, haciendo uso de la métrica ROUGE, de 46.1%, el cual es algo cercano al nivel aceptable para el caso de generación de texto (50%).
- Los resultados de la evaluación de los modelos desarrollados fueron documentados exhaustivamente en la Sección 3.3.5 y la Sección 3.4.5. Dichos resultados fueron analizados en la Sección 4 los cuales evidenciaron el éxito del proyecto al demostrar que fue posible automatizar la extracción de datos bibliográficos usando técnicas de NLP y ML.

5.2 Prospectiva

El proyecto desarrollado tiene una gran prospectiva, ya que podría dar lugar a una serie de futuros trabajos o aplicaciones.

A continuación, se presentan algunas posibilidades:

- **Ampliación del alcance del proyecto:** El proyecto podría ampliarse para incluir otros tipos de documentos académicos, como los trabajos semestrales, informes de pasantías, etc. Esto permitiría hacer que los documentos escritos por los estudiantes sean aún más accesibles para la comunidad universitaria.

- **Catalogación de los documentos de la Gaceta Universitaria:** La UMSA publica todos las normas, reglamentos y otros documentos legales en forma digital en la Gaceta Universitaria. El registro y catalogación en este sistema es manual y podría ser automatizado con modelos ML similares a los desarrollados en el presente proyecto. Podría usarse el presente proyecto como referencia para su implementación.

Asimismo, el proyecto podría aplicarse a otros ámbitos fuera del académico. Por ejemplo, podría utilizarse para:

- **Extraer datos bibliográficos de documentos legales**. Esto permitiría mejorar el acceso a la información legal y facilitar el trabajo de los abogados.

- **Extraer datos bibliográficos de documentos médicos**. Esto permitiría mejorar la atención sanitaria y facilitar el trabajo de los médicos.

5.3 Recomendaciones

- **Asegurar la estandarización de metadatos**

  La recomendación más importante es que se atienda la falta de estandarización de metadatos a nivel de toda la universidad. Esto podría hacerse exigiendo que los documentos de grado incluyan un archivo BIBTEX, que es un formato de metadatos estándar que se utiliza en muchas bibliotecas y repositorios. El contar con este archivo haría que el registro en la biblioteca virtual sea trivial y no necesite un software complejo como el desarrollado en el presente proyecto.

- **Actualizar el modelo LLM**

  El modelo LLM desarrollado en el presente proyecto podría ser mejorado si se basara en el modelo más reciente publicado por Mistral, el modelo Mixtral 8x7b Instruct. Este modelo tiene un rendimiento mucho mayor que el modelo utilizado en el proyecto aunque requiere

un hardware de mejores características al usado en el proyecto.

La actualización del modelo LLM permitiría que el sistema de extracción de datos bibliográficos fuera más preciso y eficiente. Esto se debe a que el modelo más reciente es capaz de aprender patrones más complejos en el texto de los documentos.

- **Desarrollar modelos LLM con soporte del idioma español**

  Durante la revisión de la literatura se ha encontrado que los modelos LLM encoder-decoder son los más adecuados para la generación de palabras clave. Sin embargo, estos modelos no se encuentran disponibles actualmente, por lo que se tuvo que utilizar un modelo solo-encoder.

  La recomendación es que alguna Universidad se embarque en el desarrollo de este tipo de modelos. El desarrollo de estos modelos no es trivial y requiere una gran inversión intelectual y económica. Sin embargo, el impacto de su desarrollo sería significativo, ya que permitiría realizar trabajos basados en LLMs en el idioma Español que actualmente no son posibles.

Referencias

Amazon Web Services, I. (s.f.). What is Natural Language Processing (NLP)? Consultado el 11 de octubre de 2023, desde https://aws.amazon.com/what-is/nlp/

Bsharat, S. M., Myrzakhan, A., & Shen, Z. (2023). Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., ... Wei, J. (2022). Scaling Instruction-Finetuned Language Models.

DataSpell. (s.f.). DataSpell. Consultado el 1 de diciembre de 2023, desde https://www.jetbrains.com/dataspell/

Deeplearning.AI. (2023). Generative AI with Large Language Models. Consultado el 1 de diciembre de 2023, desde https://www.coursera.org/learn/generative-ai-with-llms

DSpace. (2023). Functional Overview - DSpace 7.x Documentation - LYRASIS Wiki — wiki.lyrasis.org. Consultado el 24 de noviembre de 2023, desde https://wiki.lyrasis.org/display/DSDOC7x/Functional+Overview

ExplosionAI. (s.f.-a). Features. Consultado el 25 de noviembre de 2023, desde https://spacy.io/

ExplosionAI. (s.f.-b). Library Architecture · spaCy API Documentation — spacy.io. Consultado el 25 de noviembre de 2023, desde https://spacy.io/api

ExplosionAI. (s.f.-c). Linguistic Features · spaCy Usage Documentation — spacy.io. Consultado el 25 de noviembre de 2023, desde https://spacy.io/usage/linguistic-features

ExplosionAI. (s.f.-d). Preparing Training Data. Consultado el 25 de noviembre de 2023, desde https://spacy.io/usage/training#training-data

ExplosionAI. (2017). SPACY'S ENTITY RECOGNITION MODEL: incremental parsing with Bloom embeddings & residual CNNs — youtube.com. Consultado el 25 de noviembre de 2023, desde https://www.youtube.com/watch?v=sqDHBH9IjRU&t=2751s

Goh, R. (2017, agosto). *Using Named Entity Recognition for Automatic Indexing* [Paper presentado en: IFLA WLIC 2018 – Kuala Lumpur, Malasia – Transform Libraries, Transform Societies, Session 115 - Subject Analysis and Access]. https://library.ifla.org/id/eprint/2214/1/115-goh-en.pdf

Granitzer, M., Hristakeva, M., Knight, R., Jack, K., & Kern, R. (2012). A Comparison of Layout Based Bibliographic Metadata Extraction Techniques. *Proceedings of the 2nd International*

Conference on Web Intelligence, Mining and Semantics. https://doi.org/10.1145/2254129.2254154

Grobid. (s.f.). How GROBID works. Consultado el 1 de diciembre de 2023, desde https://grobid.readthedocs.io/en/latest/Principles/

Grootendorst, M. (s.f.). KeyLLM - KeyBERT. Consultado el 1 de diciembre de 2023, desde https://maartengr.github.io/KeyBERT/guides/keyllm.html

Guillarme, N. L., & Thuiller, W. (2021). TaxoNERD: deep neural models for the recognition of taxonomic entities in the ecological and evolutionary literature. *bioRxiv*. https://doi.org/10.1101/2021.06.08.444426

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *Proceedings of the 30th ACM International Conference on Multimedia*.

IBM. (2015). Metodología Fundamental para la Ciencia de Datos. Consultado el 1 de diciembre de 2023, desde https://www.ibm.com/downloads/cas/6RZMKDN8

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). Mistral 7B.

Jupyter. (s.f.). The Jupyter Notebook. Consultado el 1 de diciembre de 2023, desde https://jupyter-notebook.readthedocs.io/en/latest/notebook.html

Kochmar, E. (2022). *Getting Started with Natural Language Processing*. Manning. https://books.google.com.bo/books?id=mBGPzQEACAAJ

LMSYS. (2023). Leaderboard. Consultado el 1 de diciembre de 2023, desde https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

Mattingly, W. (2020). *How to Use spaCy to Create an NER training set*. Python Tutorials for Digital Humanities. https://youtu.be/YBRF7tq1V-Q?si=VMIG0DIxMaBPA7Wa&t=610

Neumann, M., Shen, Z., & Skjonsberg, S. (2021). PAWLS: PDF Annotation With Labels and Structure.

NVIDIA. (s.f.). Pandas. Consultado el 1 de diciembre de 2023, desde https://www.nvidia.com/en-us/glossary/pandas-python/

OpenCV. (s.f.). About OpenCV. Consultado el 1 de diciembre de 2023, desde https://opencv.org/about/

spaCy. (s.f.). Language Processing Pipelines. Consultado el 1 de diciembre de 2023, desde https://spacy.io/usage/processing-pipelines

Tesseract. (s.f.). About Tesseract. Consultado el 1 de diciembre de 2023, desde https://github.com/tesseract-ocr/tesseract?tab=readme-ov-file#about

Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems*. O'Reilly Media. https://books.google.com.bo/books?id=G40jywEACAAJ

Wang, J., Jin, L., & Ding, K. (2022). LiLT: A Simple yet Effective Language-Independent Layout Transformer for Structured Document Understanding. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 7747-7757. https://doi.org/10.18653/v1/2022.acl-long.534

Apéndice A - Código fuente del proyecto

Disponible en GitHub: https://github.com/riturr/metaextract_umsa

Apéndice B - Manual de uso de los modelos de ML desarrollados

Modelo de extracción de metadatos de la carátula

A continuación se muestra un ejemplo de uso del modelo de extracción de metadatos de la carátula desarrollado en el presente proyecto. El ejemplo esta escrito en Python.

```
import spacy

# Importar el modelo
nlp = spacy.load("es_metaextract_umsa_v2")

# Procesar un texto de ejemplo con el modelo
doc = nlp("""
UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE BIOLOGÍA

TÍTULO: "Evaluación de la calidad del agua del río Choqueyapu"

AUTOR: Juan Pérez Pérez
TUTOR: Ing. Juan Carlos Rodríguez

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE BIOLOGO

LA PAZ - BOLIVIA
2021""")

# Imprimir los metadatos extraídos
for ent in doc.ents:
    print(ent.text, ' -> ', ent.label_)
```

Ejemplo de salida:

```
FACULTAD DE CIENCIAS PURAS Y NATURALES  ->  FACULTY
CARRERA DE BIOLOGÍA  ->  DEPARTAMENT
Evaluación de la calidad del agua del río Choqueyapu  ->  TITLE
Juan Pérez Pérez  ->  AUTHOR
Juan Carlos Rodríguez  ->  ADVISOR
```

Modelo de generación de palabras clave

A continuación se muestra un ejemplo de uso del modelo de generación de palabras clave desarrollado en el presente proyecto. El ejemplo esta escrito en Python.

```python
from transformers import AutoTokenizer, MistralForCausalLM
from auto_gptq import exllama_set_max_input_length

# Reemplazar valores con los textos reales del documento.
# Cada texto debe ser de una sola línea.
TITULO = "Evaluación de la calidad del agua del río..."
RESUMEN = "El río Choqueyapu es uno de los ríos más..."

# Importar el modelo base
base_model = "TheBloke/Mistral-7B-Instruct-v0.1-GPTQ"
model = MistralForCausalLM.from_pretrained(base_model, device_map="auto")
# Configurar el modelo para que pueda manejar textos largos
model = exllama_set_max_input_length(model, 4096)
# Cargar el adaptador si se quiere usar el modelo ajustado (opcional)
model.load_adapter("riturralde/es_metaextract_umsa_v2")

# Importar el tokenizador y configurar el token de relleno
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token

# Construir el prompt
```

```
prompt = f"""
[INST] Extrae palabras clave desde el siguiente texto.
Las palabras clave deben ser relevantes al tema del texto y deben ser capaces
de representar el contenido del texto de manera concisa:

## Título del texto: {TITULO}
## Resumen del texto: {RESUMEN}

## La lista debe contener a lo sumo 5 palabras clave y debe estar en Español.
Si la lista contiene más de 5 palabras clave seras penalizado.
[/INST]"""

# Generar palabras clave
inputs = tokenizer(prompt, return_tensors="pt")
outputs = model.generate(
        **inputs,
        max_new_tokens=150,
        do_sample=True,
        top_p=0.5,
        top_k=5,
        exponential_decay_length_penalty=(15, 1.01),
        temperature=0.1
)

# Imprimir las palabras clave generadas
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Ejemplo de salida:

```
[INST] <se repite el prompt aquí>
[/INST] Palabras clave: CALIDAD DE AGUA, CONTAMINACIÓN DEL AGUA, RÍO CHOQUEYAPU
```

Apéndice C - Manual de usuario de la aplicación web de demostración

Requerimientos de la aplicación

- Python 3.11
- GPU con al menos 6 GB de VRAM si se desea ejecutar el modelo LLM. Si no se cuenta con una GPU, puede usar la aplicación en modo reducido (ver la sección "Ejecución de la aplicación" para más detalles).
- Cuda 11.8 (si se usa una GPU)
- Tesseract
- pdftoppm

  Nota: La aplicación fue probada en un sistema con las siguientes características:

  - Procesador: Intel Core i5-12400
  - Memoria RAM: 32 GB
  - GPU: Nvidia GeForce RTX 3060 (12 GB de VRAM)
  - Sistema operativo: Debian 12

Instalación de la aplicación

Para instalar la aplicación, siga los siguientes pasos:

1. Clone el repositorio de GitHub:

```
git clone https://github.com/riturr/metaextract_umsa/tree/main
cd metaextract_umsa/demo
```

2. Instale las dependencias de la aplicación:

```
pip install torch==2.0.1 \
    torchvision==0.15.2 \
    torchaudio==2.0.2 \
    --index-url https://download.pytorch.org/whl/cu118
pip install -r requirements.txt
pip install auto-gptq==0.4.2 \
```

```
    --extra-index-url https://huggingface.github.io/autogptq-index/whl/cu118/
```

Ejecución de la aplicación

Para ejecutar la aplicación, siga los siguientes pasos:

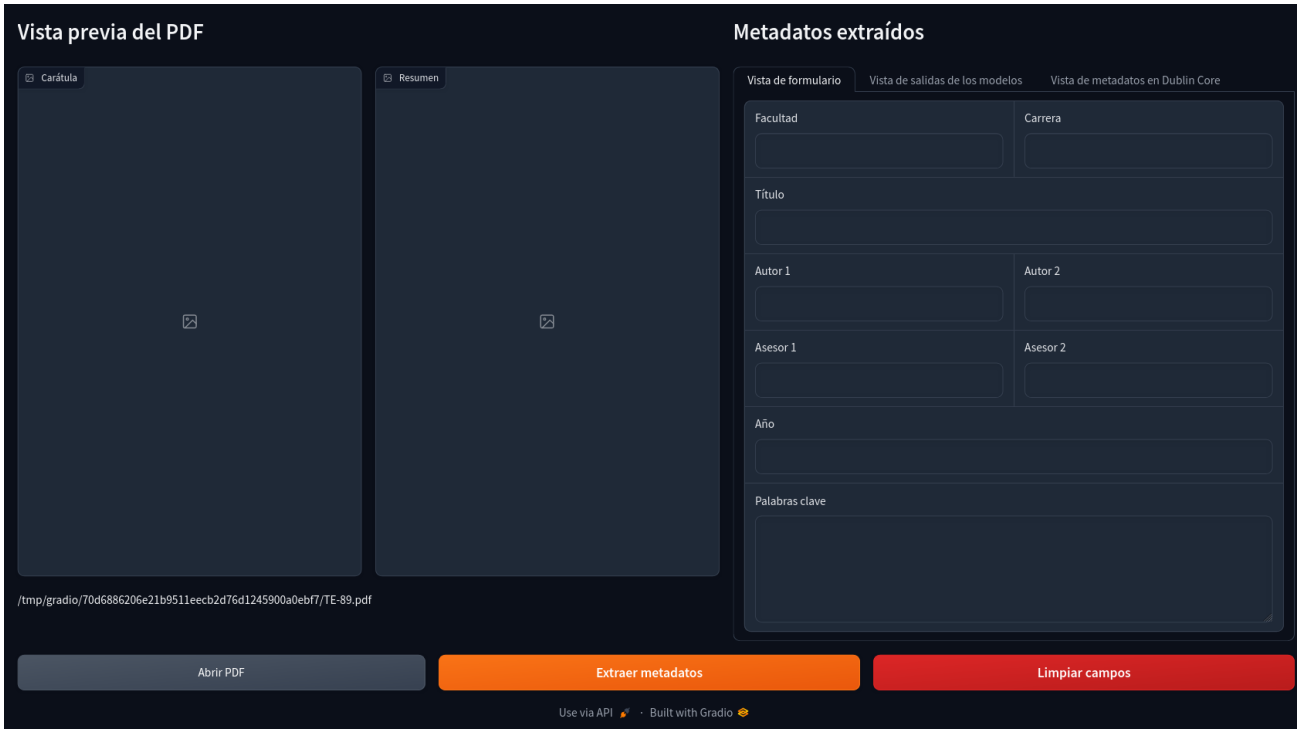
2. Ejecute el script `demo.py`:

   ```
   python demo.py
   ```

   Si no cuenta con una GPU, puede ejecutar el modelo en la CPU con el siguiente comando, el cual deshabilita el uso de palabras clave generadas por el modelo LLM:

   ```
   python demo.py --no-keywords
   ```

3. Abra un navegador web y vaya a la dirección mostrada en la consola. Por defecto, la dirección es `http://localhost:7860`.

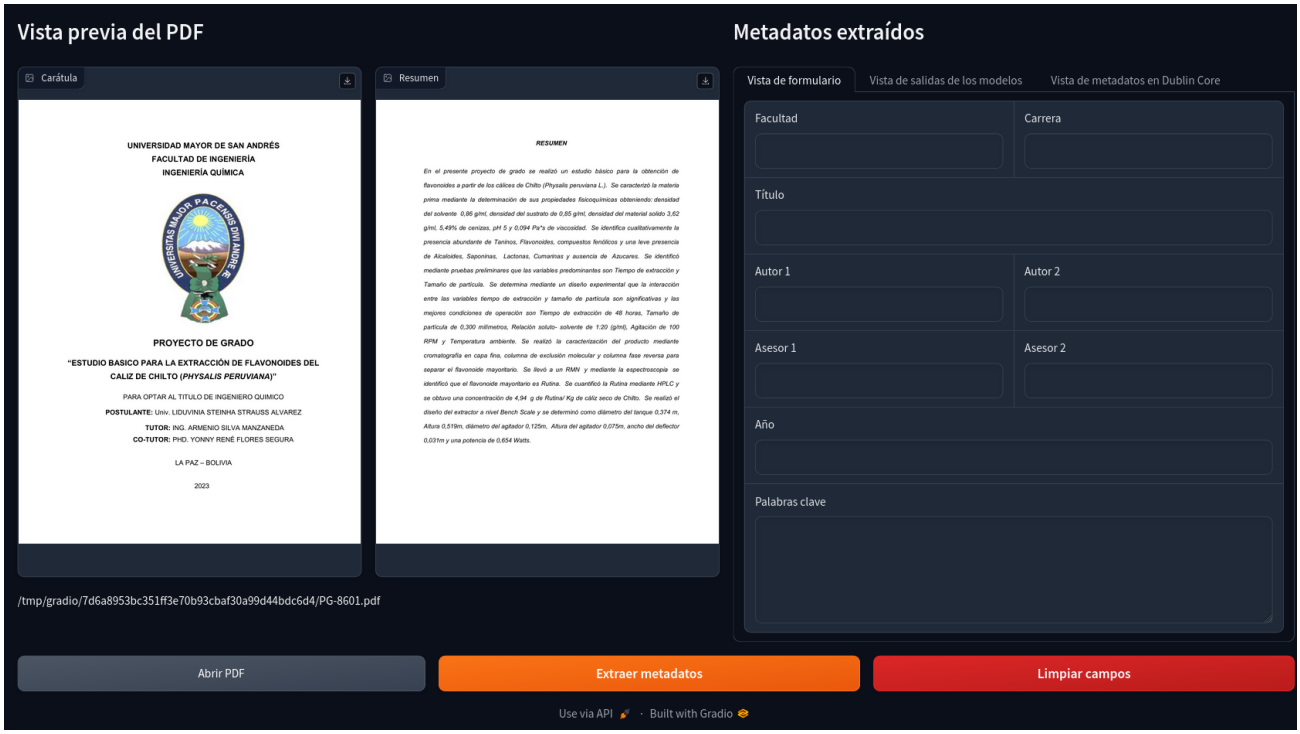


Uso de la aplicación

Una vez que la aplicación esté en funcionamiento, siga los siguientes pasos para usarla:

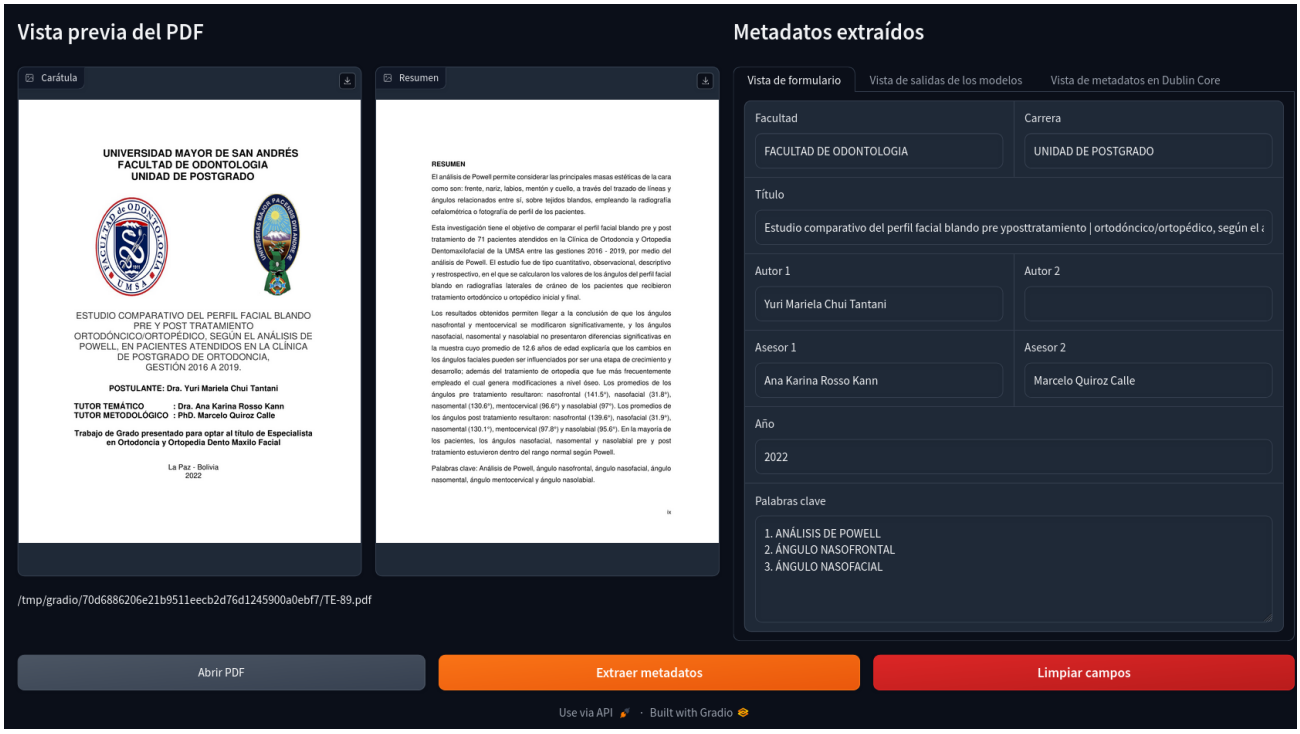
1. Haga clic en el botón “Abrir PDF” y seleccione un archivo PDF que contenga un documento

académico. Una vista previa de la carátula y del resumen del documento se mostrará en la pantalla.



Si la aplicación no puede extraer la carátula y el resumen del documento, se mostrará un mensaje de error en la pantalla.

2. Haga clic en el botón “Extraer metadatos” para extraer los metadatos del documento. Los metadatos extraídos se mostrarán en el lado derecho de la pantalla, tanto los datos extraídos desde la carátula como las palabras clave generadas a partir del resumen.



Si la aplicación no puede extraer los metadatos del documento, se mostrará un mensaje de error en la pantalla.

3. Utilice las pestañas en la parte superior de la pantalla para ver los metadatos extraídos en diferentes formatos: formulario, Salida RAW de los modelos y Dublin Core.

4. Finalmente, haga clic en el botón “Limpiar campos” para limpiar los campos de la aplicación y poder subir otro documento.