



UNIVERSIDAD DE MÁLAGA



Graduado en Ingeniería de Computadores

Extracción y generación de metadatos en textos no estructurados

Extraction and generation of metadata from unstructured texts

Realizado por

Jose Antonio Morillo Daza

Tutorizado por

Ismael Navas Delgado

Cristóbal Barba González

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, diciembre de 2022



UNIVERSIDAD
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE COMPUTADORES

EXTRACCIÓN Y GENERACIÓN DE METADATOS EN TEXTOS NO ESTRUCTURADOS

EXTRACTION AND GENERATION OF METADATA FROM UNSTRUCTURED TEXTS

Realizado por
Jose Antonio Morillo Daza

Tutorizado por
Ismael Navas Delgado
Cristóbal Barba González

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE DE 2022

Fecha defensa: enero de 2023

Resumen

Los metadatos aportan información estructurada sobre el contenido de un documento. Esta información no siempre sigue la misma estructura, se encuentra incompleta o simplemente no existe. Esto supone un problema cuando se quiere clasificar, identificar o ubicar un determinado documento. Con técnicas como la minería de textos (Text Mining) se puede automatizar la tarea de extracción de estos metadatos.

Este proyecto aporta las herramientas necesarias para extraer y generar metadatos de documentos no estructurados. Se emplean técnicas de minería de textos para conseguir dicho propósito. Los tipos de documentos serán artículos de investigación en inglés y formato PDF. Entre las funcionalidades están: lectura de documentos en formato PDF; análisis y extracción de metadatos; exportar metadatos.

Palabras clave: artículo, pdf, metadatos, text mining, Dublin Core Metadata Initiative

Abstract

Metadata provides structured information about the content of a document. This information does not always follow the same structure, is incomplete or completely missing. This is a problem when you want to classify, identify or locate a certain document. With techniques such as text mining, the task of extracting this metadata can be automated.

This project provides the necessary tools to extract and generate metadata from unstructured documents. Text mining techniques are used to achieve this purpose. The document type will be scientific papers in English and PDF format. Some of the functionalities are: reading documents in PDF format; metadata analysis and extraction; exporting metadata.

Keywords: paper, pdf, metadata, text mining, Dublin Core Metadata Initiative

Índice

1. Introducción	7
1.1. Motivación	7
1.2. Objetivos	8
1.3. Estructura	8
2. Conceptos	9
2.1. Minería de textos	9
2.2. Portable Document Format	11
2.3. Dublin Core	11
3. Herramientas	15
3.1. Anaconda	15
3.2. Spyder	16
3.3. Jupyter	16
3.4. Python	18
3.4.1. PyPDF2	18
3.4.2. pdfminer.six	19
4. Desarrollo	21
4.1. Recopilación	21
4.2. Lectura	22
4.3. Preprocesado	23
4.4. Extracción	24
4.5. Presentación	32
4.6. Pruebas	34
5. Resultados	37
6. Conclusiones	43
6.1. Líneas futuras	43

1

Introducción

1.1. Motivación

Más del 80 % de los datos que se manejan están sin estructura en la forma de texto en lenguaje natural (libros, informes, noticias, artículos de investigación, mensajes de redes sociales y páginas *web*).

En años recientes ha habido un auge en la minería de textos y el procesamiento de lenguaje natural. La minería de textos es la confluencia de procesamiento de lenguaje natural, minerías de datos, aprendizaje automático y estadística usados para extraer conocimientos de textos no estructurados.

La popularización de Internet y las tecnologías de comunicación móviles han atraído la atención en la minería de textos. Esto puede verse en campos de aplicación como la minería de opiniones y el análisis de datos médicos y financieros.

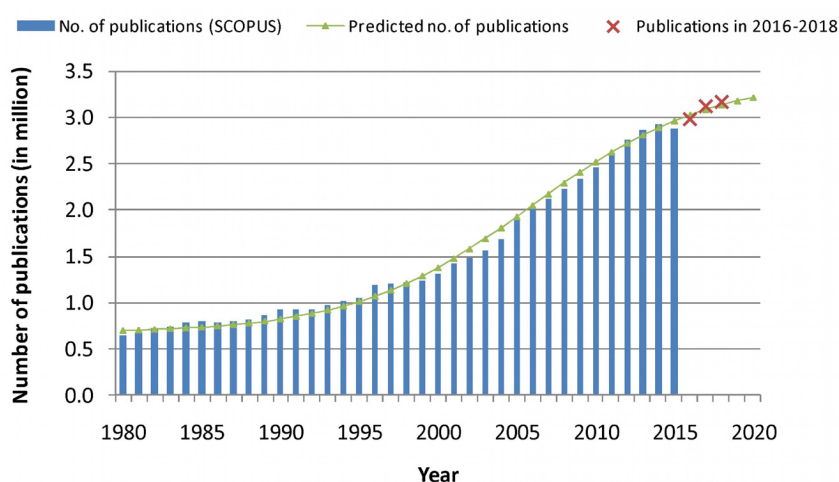


Figura 1: Publicaciones por año. [1]

Los artículos de investigación se emplean en el mundo académico para difundir los co-

nocimientos obtenidos. El número de publicaciones que se realizan anualmente no hace más que aumentar (Figura 1). Organizar o buscar un artículo en concreto para su posterior lectura requiere un tiempo valioso. Este se puede reducir si previamente se extraen los metadatos de cada uno de los artículos. De este modo se pueden filtrar los resultados para obtener así información útil.

1.2. Objetivos

El objetivo del proyecto es buscar metadatos de los artículos de investigación (papers) en formato PDF.

- El planteamiento preliminar para alcanzar el objetivo es utilizar NLP (Natural Language Processing, o procesamiento de lenguaje natural). Con esta herramienta se analiza el texto y se intenta encontrar algún patrón que encaje en alguno de los metadatos a buscar.
- Se buscarán las palabras más repetidas del texto por ser las más significativas.
- De entre los metadatos a extraer se encuentran el título y el autor.
- También se debe contabilizar el número de páginas, tablas y figuras del artículo.

1.3. Estructura

Este documento está dividido en cinco partes.

En primer lugar se encuentra esta introducción donde se establecen las bases del proyecto. A continuación, en la segunda parte, se explican los conceptos que se utilizan a lo largo de esta memoria. Le sigue la sección «Herramientas» donde se enumeran las distintas aplicaciones y librerías que se han empleado para la realización del proyecto. Posteriormente, la sección cuarta, «Desarrollo», describe los pasos necesarios para realizar el proyecto. Por último, en «Resultados» se muestran las gráficas y comentarios sobre los resultados obtenidos.

Además, se incluye un apéndice con los pasos a seguir para la instalación de Anaconda en Linux.

2

Conceptos

2.1. Minería de textos

El término minería tiene el significado de "búsqueda, descubrimiento, inducción y refinamiento". Por lo tanto, se trata de analizar textos y extraer conclusiones. El usuario espera respuestas y conclusiones a las preguntas de interés, en lugar de numerosos resultados de búsqueda, dejando que sea la máquina la que analice y encuentre respuestas. Está basada en diferentes tecnologías. Tiene su origen en técnicas como clasificación de textos, agrupación de textos y resumen automático de textos. Tiene aplicaciones en campos como la economía, servicios de información, gestión social, seguridad nacional.

Existen dos aspectos a tener en cuenta sobre el contenido del texto: no está estructurado y está descrito por el lenguaje natural, no por datos.

La minería de textos se puede clasificar en:

- Las preguntas son claras y específicas pero no se conoce respuesta.
- Se conoce el propósito general pero no se tienen preguntas definidas y específicas.

No consiste en un sistema único, si no la aplicación integrada de varias tecnologías, entre las que se pueden encontrar:

- Clasificación de textos. Es una tecnología de clasificación de patrones, es decir, clasifica un texto de acuerdo a su contenido.
- Agrupación de textos. Divide el texto en diferentes categorías. Se diferencia del método anterior en que el número de categorías no está predeterminado.
- Modelo de temas. Son modelos estadísticos que extraen temas y conceptos que se esconden tras las palabras del texto.

- Análisis emocional del texto y minería de opiniones. Se basa en información subjetiva, como puntos de vista o actitudes, dentro de un rango negativo-positivo.
- Detección y seguimiento de tema. Extrae y muestra temas desde diferentes noticias y comentarios. Especialmente interesante son los temas candentes, de los que más «se habla».
- Extracción de información. Consiste en extraer información útil y estructurada desde textos sin estructura o semiestructurados. Información como reconocimiento de entidades, desambiguación de entidades, extracción de relaciones, extracción de eventos.
- Resumen automático de textos. Genera un resumen empleando métodos de procesamiento de lenguaje natural.

Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural (en inglés Natural Language Processing, NLP) forma parte de la minería de textos y por extensión de la inteligencia artificial. Consiste en la habilidad de comprender un texto hablado o escrito del mismo modo que lo haría un ser humano.

Los datos adquiridos de diferentes fuentes no suelen tratarse directamente. Es necesaria una tarea de preprocesado. Entre los principales métodos están:

- Tokenización: Consiste en la segmentación de un texto en unidades léxicas. El inglés, por ejemplo, utiliza los espacios como separadores de palabras; solo se necesita un espacio o un signo de puntuación para realizar la separación de elementos.
- Eliminación de palabras vacías (*stop-words*): Consiste en la eliminación de palabras auxiliares, preposiciones, conjunciones y otras muy frecuentes que aportan poca información textual. Por ejemplo, en inglés son palabras como *the*, *is*, *at*, *which*, *on*. La ventaja de realizar este proceso es que se reduce el espacio de almacenamiento que se necesita y se mejora la eficacia de extracción de información.
- Normalización de palabras: Consiste en fusionar las diferentes variaciones o formas en las que pueden aparecer las palabras. Así se consigue mejorar la eficiencia del procesamiento del texto. Hay dos modos de realizarlo. Uno es la lematización (*lemmatization*), la restauración de palabras en sus formas originales. Por ejemplo, se convierte *cats* en

cat o *did* en *do*. El otro es la extracción de la raíz (*stemming*), que es la eliminación de afijos para obtener las raíces de las palabras. Estas no tienen por qué tener un significado semántico. En inglés, por ejemplo, se pasa de *fisher* a *fish* y de *effective* a *effect*.

Un modelo lingüístico de gran utilidad en el procesamiento de lenguaje natural es la creación de n-gramas. La idea es crear un fragmento de texto compuesto por n palabras adyacentes, donde n es mayor o igual a dos. La unidad básica se denomina monograma en lugar de palabra. Por lo tanto, cuando se trabaja con dos palabras se habla de bigramas; y con tres palabras, trigramas.

2.2. Portable Document Format

Portable Document Format o PDF es un formato de fichero que se emplea para representar documentos. Fue creado por Adobe Systems Incorporated en 1993. Su uso está bastante extendido por la facilidad que ofrece para visualizar e intercambiar documentos, independientemente del entorno en el que fueron creados o en el que serán visualizados.

Un documento PDF puede incluir metadatos. Estos se pueden almacenar de dos formas distintas: como un diccionario de información del documento (Document Information Dictionary) o como flujo de metadatos (Metadata Stream)

El diccionario de información del documento era forma original de inclusión de metadatos. A partir de la versión 1.4, se añadió el flujo de metadatos. De los dos métodos, que pueden convivir en el mismo fichero, el segundo es el preferido.

2.3. Dublin Core

Dublin Core es un estándar de representación de metadatos para la descripción de recursos. Consiste en un conjunto de quince elementos (Figura 2). Cada uno de ellos posee un nombre bastante descriptivo de modo que sea más comprensible su contenido.

Los quince elementos son los siguientes:

contributor

Es la persona u organización que ha hecho contribuciones intelectuales significativas al recurso. Su contribución es secundaria, no se especifica en la etiqueta creator. Se utiliza por ejemplo, para el traductor o el editor en el caso de los libros.

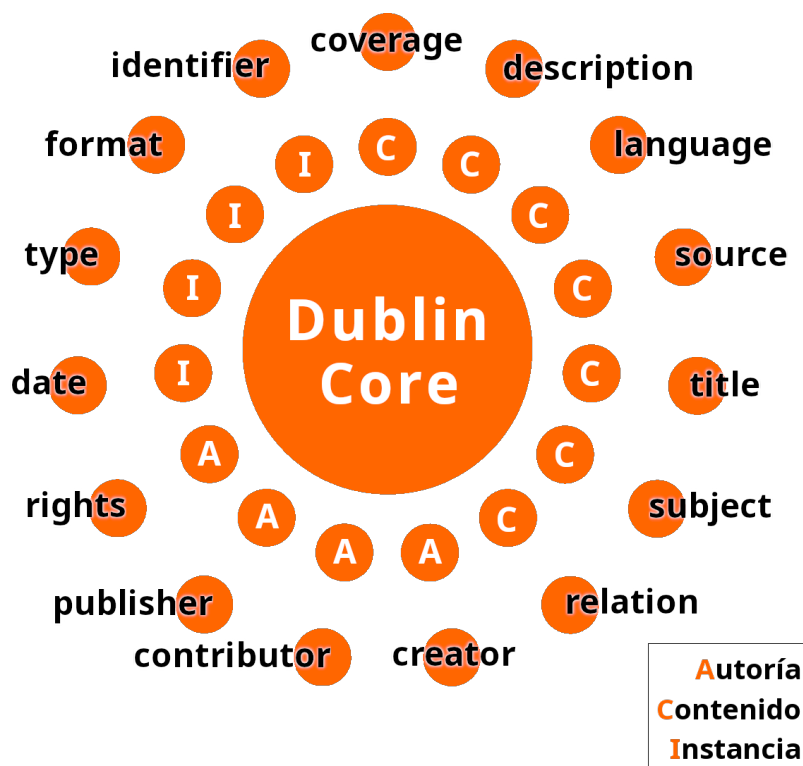


Figura 2: Logo de Dublin Core con sus quince elementos.

coverage

Son las características espaciales o temporales del contenido intelectual del recurso. La cobertura espacial se refiere a una región física (por ejemplo, un sector celeste) utilizando nombres de lugares o coordenadas (por ejemplo, longitud y latitud). La cobertura temporal se refiere al tema del recurso y no a la fecha de su creación o puesta a disposición (esta última pertenece al elemento fecha). La cobertura temporal se suele especificar utilizando períodos de tiempo con nombre (por ejemplo, el Neolítico) o el mismo formato de fecha/hora que se recomienda para el elemento.

creator

Es la persona u organización responsable (principalmente) de la creación del contenido intelectual del recurso. Se utiliza por ejemplo, para los escritores que firman un libro.

date

Es la fecha asociada a un evento del recurso. Normalmente la fecha de publicación o crea-

ción. Se recomienda utilizar la norma ISO 8601, que tiene el formato AAAA y AAAA-MM-DD, siendo AAAA el año, MM el mes y DD el día.

description

Es la descripción textual del contenido del recurso. Suele ser un resumen del contenido en el caso de recursos escritos.

format

Es el formato de los datos. Puede incluir información extra como las dimensiones o la duración. Tiene su utilidad para identificar las herramientas necesarias para poder visualizar el recurso.

identifier

Es un texto o número utilizado para identificar de forma inequívoca el recurso. Por ejemplo, el ISBN en el caso de los libros o el DOI para los artículos científicos.

language

Es el idioma del contenido intelectual del recurso. Se recomienda utilizar los códigos de idioma IETF, recogidos en la norma RFC 1766.

publisher

Es la entidad responsable de que el recurso esté disponible en su forma actual, como una editorial o un departamento universitario.

relation

Es un identificador de un recurso secundario y su relación con el recurso actual. Este elemento se usa para vincular recursos relacionados.

rights

Es una declaración de gestión de derechos. Suele enlazar con información sobre los derechos de uso del recurso.

source

Es la fuente de la que se deriva el recurso actual. Solo se utiliza cuando este recurso secundario se considere relevante para el descripción del recurso presente.

subject

Es el asunto, tema o materia que define el recurso. Se suele expresar con palabras clave o frases descriptivas. Para facilitar la interoperatividad se utilizan vocabularios controlados o sistemas de clasificación formales.

title

Es el título dado al recurso, normalmente por parte del autor o del editor.

type

Es la categoría del recurso. Por ejemplo, una novela, un ensayo o informe técnico.

3

Herramientas

3.1. Anaconda

Anaconda¹ es una distribución de software que incluye diversas aplicaciones y librerías, entre las cuales están Python, Spyder y Jupyter. Tiene como objetivo facilitar la administración e implementación de librerías y paquetes, además de simplificar la gestión de entornos. Es la plataforma más completa para trabajar con Data Science.

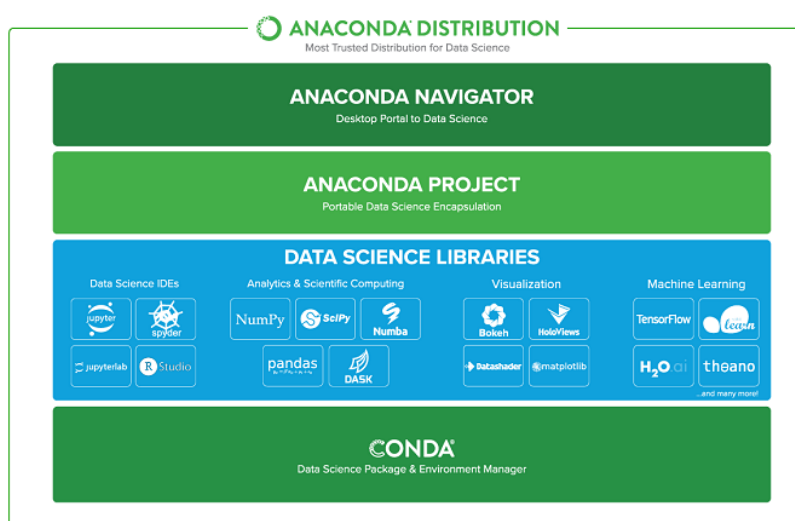


Figura 3: Elementos principales de Anaconda. [2]

Se clasifica en cuatro partes: Anaconda Navigator, Anaconda Project, Conda y las librerías de ciencia de datos (Figura 3).

La versión que se ha empleado para la realización del proyecto es la 2022.05 para Linux. Fue publicada el 10 de mayo de 2022. En el **Apéndice A** se explica con pasos todo el proceso de instalación.

¹<https://www.anaconda.com/>

3.2. Spyder

Spyder^{II} es un entorno de desarrollo integrado para Python. Es gratuito y de código abierto. Fue desarrollado por Pierre Raybaut a finales de los noventa. Está escrito en Python. Su diseño está enfocado para uso científico y análisis de datos.

Algunas de las características más importantes son:

- Editor de texto. Tiene resaltado de código y autocompletado.
- Consola interactiva. Permite probar y depurar el código en tiempo real. También permite trazar gráficos en línea; así se facilita
- Gestión de proyectos. Posee un explorador de archivos e inspector de objetos y variables.
- Integración con librerías científicas. Se integra con librerías como NumPy, Pandas, SciPy y Matplotlib.
- Visor de documentación. Puede mostrar la documentación de cualquier clase o función.
- Soporte multiplataforma. Está disponible en Windows, Linux y MacOS.
- Análisis de código. Permite detectar problemas de estilo, malas prácticas, posibles errores y otros problemas del código.

La distribución Anaconda utilizada para el proyecto incluye la versión 5.1.5 de Spyder.

3.3. Jupyter

Jupyter Notebook^{III} es una aplicación web para crear y compartir documentos computacionales. Ofrece un manejo sencillo, y está centrado en los documentos. Los cuadernos (*notebooks*) pueden contener código y datos.

JupyterLab es una evolución del anterior. Su interfaz permite a los usuarios configurar y organizar flujos de trabajo en ciencia de datos. Posee un diseño modular que invita a realizar ampliaciones que mejoren su funcionalidad.

^{II}<https://www.spyder-ide.org/>

^{III}<https://jupyter.org/>

Un cuaderno consiste en una serie de celdas. Pueden contener código, texto o visualizaciones. La celda de código, al ejecutarse, genera una nueva celda, situada debajo, con el resultado de la ejecución. Las visualizaciones de datos se realizan gracias a la integración de librerías gráficas como Matplotlib. El texto se escribe en formato Markdown^{IV}, que es un lenguaje de marcado.

A continuación se muestra cómo se vería el clásico «Hello World!» en un cuaderno Jupyter. En la figura 4 están las tres celdas que componen el cuaderno. La primera y tercera están definidas como Markdown y la segunda como IPython. En la figura 5 se vuelve a mostrar el mismo cuaderno tras ejecutar el código.

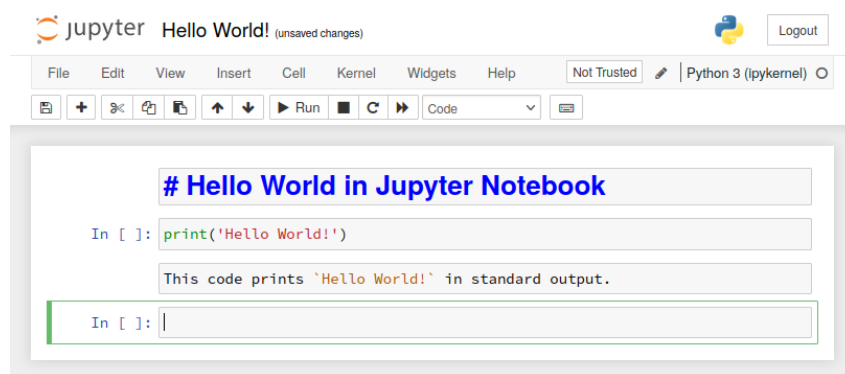


Figura 4: Código de «Hello World!» en Jupyter

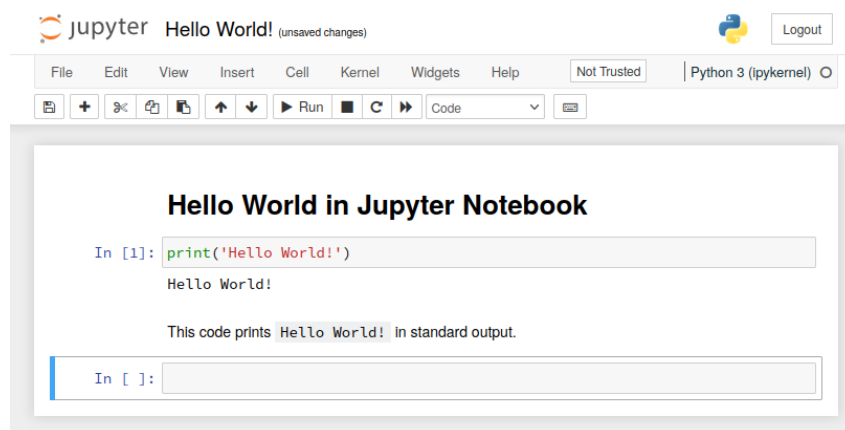


Figura 5: Ejecución de «Hello World!» en Jupyter

En la distribución Anaconda usada se incluyen las versiones 3.3.2 de JupyterLab y 6.4.8 de Notebook.

^{IV}<https://daringfireball.net/projects/markdown/>

3.4. Python

Python^v es un lenguaje de programación de propósito general. Fue creado a finales de los años 80 por Guido van Rossum. Entre sus características destacadas están:

- Es interpretado. El código es procesado en tiempo de ejecución por el intérprete.
- Es orientado a objetos. Permite estructurar los programas de forma modular.
- Es dinámico. Las variables pueden ser de cualquier tipo, ya que de base todas las variables son objetos.
- Es multiplataforma. Puede ser ejecutado en cualquier plataforma que tenga instalado el intérprete.
- Es de código abierto.

Es conocido por su sencillez, facilidad de uso y por poseer gran cantidad de librerías. Es una opción tanto para principiantes como para desarrolladores experimentados. Es muy legible, con una sintaxis clara y concisa que facilita la comprensión y la escritura de código.

En los últimos años ha aumentado su popularidad en el desarrollo de *software* para la ingeniería de datos y *machine learning*.

La versión usada para el proyecto es la 3.9, que viene integrada en la distribución Anaconda.

3.4.1. PyPDF2

PyPDF2^{vi} es una librería de Python cuya función es trabajar con los archivos PDF. Puede realizar una serie de tareas relacionadas con estos documentos, tales como:

- Extraer texto y metadatos
- Fusionar y dividir documentos
- Rotar, recortar y reordenar páginas
- Cifrar y descifrar archivos

^v<https://www.python.org/>

^{vi}<https://github.com/py-pdf/PyPDF2>

- Añadir y eliminar anotaciones y campos de formulario

La versión instalada de la librería PyPDF2 es la 2.10.4.

3.4.2. **pdfminer.six**

Pdfminer.six^{vii} es una librería de Python que sirve para extraer texto, imágenes y otra información de documentos PDF. Es una versión mantenida por la comunidad de la librería PDF-Miner. Una de sus principales ventajas es su capacidad para extraer texto con gran precisión, incluso de documentos con diseños y formatos complejos. Sus características son:

- Escrito íntegramente en Python.
- Examina, analiza y convierte documentos PDF.
- Extrae contenido como texto, imágenes, html o hOCR.
- Soporta la especificación PDF-1.7.
- Soporta idiomas y *scripts* de escritura vertical.
- Soporta varios tipos de fuentes (Type1, TrueType, Type3 y CID).
- Soporta imágenes (JPG, JBIG2, Bitmaps).
- Soporta varios tipos de compresión (ASCIISHexDecode, ASCII85Decode, LZWDecode, FlateDecode, RunLengthDecode, CCITTFaxDecode).
- Soporta encriptado RC4 y AES.
- Soporta la extracción interactiva de formularios AcroForm.
- Extracción de índices.
- Extracción de contenidos etiquetados.
- Análisis automático del maquetado.

La versión instalada de la librería pdfminer.six es la 20220524.

^{vii}<https://github.com/pdfminer/pdfminer.six>

4

Desarrollo

El proyecto se divide en varias fases: recopilación de artículos de investigación, lectura del contenido, preprocesado del contenido, extracción de información y presentación del resultado en XML.

En primer lugar se recopilan todas las muestras a tratar. Estas deben situarse en una misma carpeta.

Sigue una fase de lectura en la que se extrae el contenido utilizando las librerías PyPDF2 (para metadatos) y pdfminer.six (para el contenido).

La siguiente fase es el preprocesado de la información. Aquí el proceso se divide, pues se prepara la información de distinta manera según sea para trabajar con los metadatos del PDF o su contenido.

Con los datos ya preparados se inicia la fase de extracción. Se usarán distintas técnicas según se trabaje con el texto plano o con los bloques de texto.

Por último, se exportan todos los metadatos a un XML. Estos siguen el estándar Dublin Core.

En las siguientes secciones se verán con detalle todas estas fases. El flujo de información sigue el esquema de la figura 6.

4.1. Recopilación

Para realizar tanto las pruebas como para obtener unos resultados admisibles, se hizo una recopilación de artículos de investigación de PDF. En total se obtuvieron cien artículos de diferentes materias aunque de predominancia tecnológica.

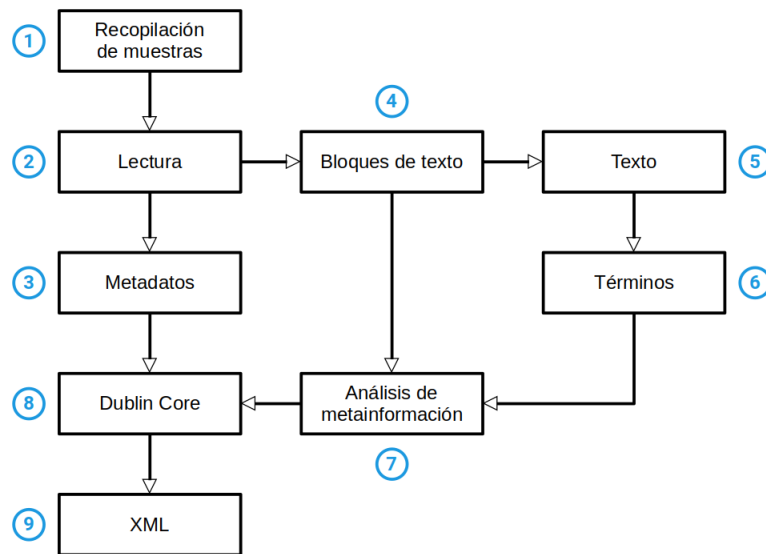


Figura 6: Fases de desarrollo.

4.2. Lectura

En primer lugar se leen los metadatos utilizando la librería PyPDF2. De este modo se consigue obtener una base de metainformación de la que partir.

El documento se lee utilizando la clase PdfFileReader. El objeto generado tiene información de los tipos de metadatos que contiene el PDF. Estos se encuentran en los objetos metadata (de tipo DocumentInfo) y xmp_metadata (de tipo XmpMetadata). Estos metadatos necesitan un filtrado previo. Aquellos valores que estén vacíos o que solo contengan espacios en blanco, se eliminan.

Los valores contenidos en DocumentInfo se devuelven como cadena de texto. En el caso del autor se hace una separación de los nombres mediante expresiones regulares. Se parten teniendo en cuenta que la mayoría de los casos con varios autores estos están separados por comas o puntos y comas.

Los valores contenidos XmpMetadata se devuelven con varios tipos según el campo. Se han reducido a tres casos:

- cadenas de texto para los campos dc_coverage, dc_format, dc_identifier y dc_source
- listas de textos para los campos dc_contributor, dc_creator, dc_description, dc_language,

dc_publisher, dc_relation, dc_rights, dc_subject, dc_title y dc_type

- lista de fechas para el campo dc_date

El campo dc_title se transforma de diccionario a lista. Se da preferencia en el orden de la lista a los títulos en inglés o que estén marcados como default (por defecto). El resto de títulos, si son distintos, se añaden al final.

El campo dc_creator se le realiza una limpieza de cadenas vacías o que solo contengan espacios en blanco.

Después se lee el contenido utilizando la librería `pdfminer.six`. Esta fase se realiza en dos pasos. Por un lado se obtiene todo el contenido como un único texto; y por otro, todos los bloques de texto.

Primeramente se lee el contenido por bloques. Un bloque de texto es un conjunto de líneas de texto que tienen un margen mínimo de separación. Los párrafos que tienen un margen de separación mayor, conforman otro bloque. Esta fase supone ya un preprocesado de información, que se utilizará posteriormente en la extracción de algunos metadatos como el título.

En segundo lugar, se lee todo el contenido como texto plano. Para tal fin, se recorren todos los bloques y se unen con saltos de línea. Este texto servirá más adelante para procesarlo mediante NLP.

4.3. Preprocesado

El preprocesado consiste en una serie de pasos para preparar el contenido del PDF que más tarde se usará para analizarlo o procesarlo. Durante la lectura se van recorriendo los bloques de texto y se van procesando y almacenando en objetos de la clase PdfBlock.

```
1 @dataclass
2 class PdfBlock:
3     page_number: int
4     block_number: int
5     text: str
6     top_margin_percent: float
7     left_margin_percent: float
```



```
8     font_sizes: dict[int, float]
9     font_bold_rate: float
10    font_italic_rate: float
```

Cada objeto almacena:

- El número de página al que pertenece el bloque.
- El número de bloque del documento.
- El texto del bloque, al que se le han limpiado los espacios sobrantes.
- Los márgenes superior e izquierdo del bloque, en porcentaje respecto al tamaño de página. Estos márgenes se guardan como porcentaje para evitar la influencia del tamaño de página a la hora de tener en cuenta los valores.
- Un diccionario con los tamaños de letra y en que proporción. Se contabilizan los tamaños de letra y se dividen por la longitud del texto.
- El ratio de letras del bloque que están en negrita. Se cuentan las letras que están en negrita y se dividen por el número de letras totales.
- El ratio de letras del bloque que están en cursiva. Se cuentan las letras que están en cursiva y se dividen por el número de letras totales.

El texto total del documento, obtenido de la suma de los textos de los bloques, se utiliza como entrada de la clase PdfNlp. Esta parte del preprocesado hace uso de la librería `nltk`.

La clase guarda un listado de todas las palabras (propiedad `word_tokens`). También se generan todas las palabras en minúscula, filtrando aquellas que no contengan letras (propiedad `word_topics`). Con estas dos listas se pueden calcular los n-gramas, raíces o lemas del documento. Para ello se dispone de métodos de clase. Por ejemplo, para calcular los bigramas más frecuentes se puede usar el método `bigram_freq`.

4.4. Extracción

La extracción de información se realiza mediante el análisis del contenido del documento. Dependiendo del metadato, el método empleado difiere de un caso a otro.

Título

Este metadato es el más identificativo de un documento. Independientemente que las palabras clave o el resumen puedan aportar una información más extensa del contenido, el título ofrece de un rápido vistazo una idea general del tema que se va a tratar en el artículo de investigación.

A diferencia de otros metadatos, este es más habitual que se encuentre definido en los metadatos del PDF. En la sección Resultados (Figura 16) se muestra una gráfica con los conteos de los campos rellenos.

A Data Provenance based Architecture to Enhance the Reliability of Data Analysis for Industry 4.0

Peng Li¹
¹Ostwestfalen-Lippe University of Applied Science
inIT - Institute industrial IT
Lemgo, Germany
Email: {peng.li, oliver.niggemann}@hs-owl.de

Oliver Niggemann^{1,2}
²Fraunhofer IOSB-INA
Application Center Industrial Automation
Lemgo, Germany
Email: {oliver.niggemann}@iosb-ina.fraunhofer.de

Abstract—Integrating data analysis into workflows is a recent tendency in manufacturing sectors. According to the vision of Industry 4.0, data analysis can be automatically performed at any point of workflows if needed. In distributed and complex manufacturing systems, checking the integrity of data analysis processes is becoming more and more challenging and the dependency between (intermediate) analysis results is no more

which gets data as input, then gives out the processed data as result. Therefore, data analysis processes can be viewed as processes of data creation and data propagation along time, which is named as data provenance. Although data provenance has been increasingly used in various applications [2], only few works have been carried out for applying data provenance in

Figura 7: Muestra del título un artículo

Basándose en la revisión y análisis de los diferentes artículos, se observa que en la mayoría de ellos el título tiene una tipografía mayor que el resto (Figura 7). También se observa que la tipografía está en negrita, de forma que destaque de otros párrafos (Figura 8).

Además, si se tiene en cuenta la situación espacial del bloque de texto, se observa que los títulos se encuentran al inicio de la primera página (Figura 9).

Teniendo en cuenta todos estos factores se procede a realizar el procesamiento de los bloques de texto.

En primer lugar se recopilan todos los bloques de texto de la primera página. De todos estos bloques se calcula el máximo tamaño de letra.

Posteriormente, se filtran todos los bloques que tengan el tipo de letra igual al máximo obtenido.

Si el número de bloques es solo uno, se selecciona el texto del bloque. En otro caso, se

WINGS: Intelligent Workflow-Based Design of Computational Experiments

Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro Antonio González-Calero,
Paul Groth, Joshua Moody, Ewa Deelman

Information Sciences Institute
University of Southern California

Abstract

Scientists use computational experiments to study natural phenomena through the lens of software tools and computer programs. A computational experiment consists of a description of how selected datasets are to be processed by a series of software components, in what order, and with what parameter configurations. Designing computational experiments is a challenging task for scientists because of the many

Figura 8: Muestra del título un artículo

realiza un segundo filtrado. Se seleccionan los bloques que tengan la tipografía en negrita. Al igual que en el caso anterior, si es uno, se selecciona el texto como resultado parcial.

```
1 def __filter_text_blocks(self, text_blocks: list) ->
    list:
2     max_font_size = self.__max_font_size(text_blocks)
3     result_blocks = [b for b in text_blocks if b.
        font_size() == max_font_size
4         and b.is_bold() and b.top_margin_percent <
        50]
5     if not result_blocks:
6         result_blocks = [b for b in text_blocks if b.
            font_size() == max_font_size]
7     return result_blocks
```

Si el número de bloques es mayor que uno, entonces se concatenan los textos bloques separados por un espacio.

Finalmente, y antes de devolver el resultado, se realiza una limpieza de espacios en blanco y saltos de línea. El texto devuelto lo formará una sola línea de texto.

RESEARCH ARTICLE

A metadata approach to evaluate the state of ocean knowledge: Strengths, limitations, and application to Mexico

Juliano Palacios-Abrantes¹*, Andrés M. Cisneros-Montemayor¹, Miguel A. Cisneros-Mata², Laura Rodríguez³, Francisco Arreguín-Sánchez⁴, Verónica Aguilar⁵, Santiago Domínguez-Sánchez⁶, Stuart Fulton⁷, Raquel López-Sagástegui⁸, Héctor Reyes-Bonilla⁹, Rocio Rivera-Campos⁹, Silvia Salas^{10,11}, Nuno Simoes^{12,13,14}, William W. L. Cheung¹



OPEN ACCESS

Citation: Palacios-Abrantes J, Cisneros-Montemayor AM, Cisneros-Mata MA, Rodríguez L,

1 Institute for the Oceans and Fisheries, The University of British Columbia, Vancouver, British Columbia, Canada, 2 Instituto Nacional de Pesca y Acuicultura, Guaymas, Sonora, México, 3 Environmental Defense Fund de México, La Paz, Baja California Sur, México, 4 Instituto Politécnico Nacional, Centro Interdisciplinario de Ciencias Marinas, La Paz, Baja California Sur, México, 5 Comisión Nacional para el Conocimiento y Uso de la Biodiversidad, Ciudad de México, México, 6 University of California San Diego, Scripps Institution of Oceanography, La Jolla, California, United States of America, 7 Comunidad y Biodiversidad, Cancún, Quintana Roo, México, 8 Universidad Autónoma de Baja California Sur, La Paz, Baja California Sur, México, 9 SmartFish Rescate de Valor, A.C., La Paz, Baja California Sur, México, 10 Instituto Politécnico Nacional, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, Mérida, Yucatán, México, 11 Universidad Marista de Mérida, Mérida, Yucatán, México, 12 Universidad Nacional Autónoma de México, Unidad Multidisciplinaria de Docencia e Investigación – Sisal, Yucatán, México, 13 Laboratorio Nacional de Resiliencia Costera, Laboratorios Nacionales, Ciudad de México, México, 14 Texas A&M University, Corpus Christi, Texas, United States of America

Figura 9: Muestra del título un artículo

Autor

Los métodos probados para extraer el autor no dan resultados. Estos se encuentran en posiciones distintas, mezclados con otros datos (cargo, entidad, ciudad) y símbolos (referencias).

Se partió de la base que los autores aparecen en la primera página del documento. Por lo tanto, solo se analizaron los bloques de texto de esta página.

El primer paso que se siguió fue dividir el bloque de texto en *tokens* (palabras). Para ello se utilizó etiquetado gramatical (POS tagging). Esta técnica consiste en separar cada una de las palabras del texto según su función en el mismo: artículo, sustantivos, verbo, etc. Se buscó aquellos bloques donde se encontraran sustantivos en mayor proporción.

Después, se volvía a analizar este bloque utilizando reconocimiento de entidades nombradas (NER). Esta técnica consiste en identificar a qué categoría pertenece el nombre: si se trata de personas, lugares, cantidades, expresiones de tiempo, etc. Aquel bloque con mayor porcentaje de personas, se consideraba que correspondía a una lista de autores.

Por último, se separaban los nombres utilizando expresiones regulares. Buscando comas o puntos y coma, se dividía el bloque en porciones. Cada porción, al final del proceso, debería representar a un autor.

Algunos de los problemas fueron a la hora de identificar el bloque correcto. Al no seguir un patrón estandarizado, no se conseguía localizar. En otros casos, cuando se trata de aplicar

el reconocimiento de entidades no se consigue establecer diferenciación entre los elementos. El factor de acierto no fue el esperado.

Descripción

El campo descripción se extrae de los metadatos del PDF. Si este se encuentra vacío se procede al análisis del texto.

Para rellenar este metadato se utiliza el resumen (*abstract*) que poseen los artículos de investigación (Figura 10). Este elemento se encuentra en la primera página, antes que cualquier otro contenido.

Abstract—Wearable medical sensors (WMSs) are garnering ever-increasing attention from both the scientific community and the industry. Driven by technological advances in sensing, wireless communication, and machine learning, WMS-based systems have begun transforming our daily lives. Although WMSs were initially developed to enable low-cost solutions for continuous health monitoring, the applications of WMS-based systems now range far beyond health care. Several research efforts have proposed the use of such systems in diverse application domains, e.g., education, human-computer interaction, and security. Even though the number of such research studies has grown drastically in the last few years, the potential challenges associated with their design, development, and implementation are neither well-studied nor well-recognized. This article discusses various services, applications, and systems that have been developed based on WMSs and sheds light on their design goals and challenges. We first provide a brief history of WMSs and discuss how their market is growing. We then discuss the scope of applications of WMS-based systems. Next, we describe the architecture of a typical WMS-based system and the components that constitute such a system, and their limitations. Thereafter, we suggest a list of desirable design goals that WMS-based systems should satisfy. Finally, we discuss various research directions related to WMSs and how previous research studies have attempted to address the limitations of the components used in WMS-based systems and satisfy the desirable design goals.

Figura 10: Sección «Abstract» en un artículo de investigación

Para localizarlo se busca un bloque de texto que empiece por «abstract» y se almacena todo el párrafo que le sigue. En algunos casos, el título «abstract» se encuentra en un bloque que precede (Figura 11). Entonces se busca el bloque siguiente y se guarda.

Se dan casos que no se extrae el contenido correctamente. Cuando el resumen es muy extenso y ocupa más de una página. También cuando el resumen se encuentra dividido en secciones, que emplean cabeceras que se repiten en el resto del documento. Por ejemplo, un *abstract* que contenga un apartado «Introduction» (Figura 12) y unos párrafos después se encuentre una sección de igual nombre.

Asunto

El asunto, como ocurre en los demás casos, se extrae primeramente del XMP del PDF. Se limpian los espacios sobrantes. Si se encuentra vacío entonces se procede a analizar el texto. Hay dos opciones adicionales para su completado: expresiones regulares o términos más frecuentes.

En el primer caso, se emplea un método similar al que se utiliza con «description». Se busca en la primera página del documento un bloque que comience por el texto «keywords» (Figura

Abstract

Introduction

The UK Biobank (UKB) is a resource that includes detailed health-related data on about 500,000 individuals and is available to the research community. However, several obstacles limit immediate analysis of the data: data files vary in format, may be very large, and have numerical codes for column names.

Results

ukbtools removes all the upfront data wrangling required to get a single dataset for statistical analysis. All associated data files are merged into a single dataset with descriptive column names. The package also provides tools to assist in quality control by exploring the primary demographics of subsets of participants; query of disease diagnoses for one or more individuals, and estimating disease frequency relative to a reference variable; and to retrieve genetic metadata.

Conclusion

Having a dataset with meaningful variable names, a set of UKB-specific exploratory data analysis tools, disease query functions, and a set of helper functions to explore and write genetic metadata to file, will rapidly enable UKB users to undertake their research.

Figura 11: Sección «Abstract» con cabecera

13). El texto que le continúa hasta el final se consideran que son las palabras claves divididas por un separador. Los separadores habituales son la coma y el punto y coma. Para dividir cada *keyword* se utilizan expresiones regulares, que además limpian los espacios sobrantes. Se almacena entonces cada una de las palabras clave en una lista.

En el segundo caso, se emplea una técnica de procesamiento de lenguajes naturales, que son los bigramas. Se trata de calcular la frecuencia de dos palabras seguidas a lo largo del texto. Se ha seleccionado bigramas en lugar de palabras más frecuentes porque aportan más información. Y bigramas en lugar de trigramas porque estas son menos frecuentes en el texto.

Los bigramas se obtienen separando el texto en unidades básicas (tokens). Con la función `word_tokenize` se obtiene una lista de palabras. Estas palabras se convierten a minúsculas, para evitar variaciones de la misma. De esta lista se eliminan las *stop words*, que son palabras frecuentes en el lenguaje pero que no aportan significado, como por ejemplo, los artículos o conjunciones. También se eliminan todos los números y símbolos. Esta lista de palabras ya filtradas, se pasa a la función `bigrams` de la librería `nltk` que devuelve una lista de parejas de palabras (bigramas). De esta lista se extraen los bigramas más frecuentes, que serán utilizados como «keywords».

Abstract

This paper presents the formal syntax and the operational semantics of Taverna, a workflow management system with a large user base among the e-Science community. Such formal foundation, which has so far been lacking, opens the way to the translation between Taverna workflows and other process models. In particular, the ability to automatically compile a simple domain-specific process description into Taverna facilitates its adoption by e-scientists who are not expert workflow developers. We demonstrate this potential through a practical use case.

Figura 12: Sección «Abstract» con apartados

Keywords: feature extraction; attention mechanism; pyramid network; deep learning; text detection

Figura 13: Sección «Keywords» en un artículo de investigación

Utilizando estos tres métodos de extracción de información, se consigue que el campo «subject» nunca esté vacío.

Idioma

El idioma no es un campo de información especialmente útil para este estudio en concreto, dado que todos los artículos que se analizan deben estar en inglés.

Se ha incluido como medio de verificación, en caso de que se le proporcione un documento en otro idioma que no sea el inglés.

El valor que se toma por defecto es el que se incluye en el metadato del XMP del PDF. En caso de que se encontrara vacío, lo más habitual, se procede a detectar el idioma del contenido del documento.

Para identificarlo se utiliza la librería `langdetect`. Esta proporciona una interfaz bastante sencilla. Se le pasa un texto y devuelve el código ISO 639-1 del idioma con mayor probabilidad.

```
1 def language(self) -> str:
2     result = detect(self.text())
```

```

3     if result != 'unknown':
4         return result
5     else:
6         return ''

```

En caso de no haber una coincidencia positiva, devuelve el valor «unknown». Cuando esto suceda, no se rellenará el campo language en los metadatos Dublin Core.

Número de páginas

El cálculo del número de páginas se realiza directamente usando la librería PyPdf2. Esta separa el contenido del documento por páginas. El proceso es tan sencillo como calcular la longitud de dicha secuencia.

```

1     self.num_pages: int = len(reader.pages)

```

Número de tablas

Para contabilizar el número de tablas se hizo un primer intento con la librería pdfminer.six pero no se encontró ningún método que facilitara la identificación de una tabla en el cuerpo del PDF.

Posteriormente se investigó la posibilidad de emplear la librería tabula, pero está dedicada a extraer el contenido, no a su localización o contabilización.

Finalmente, se decide emplear expresiones regulares. Dado que las tablas deben llevar un pie identificativo, se localizan examinando el texto. Cualquier bloque texto que comience por la palabra «table» y se continúe por un número se considera que sucede a una tabla en el documento. Para evitar falsos positivos o tablas que se extiendan más de una página, se almacenan los números de tabla en un conjunto. Así se consigue que al contabilizar, se haga con identificadores de tabla únicos.

Número de imágenes

Otra de las estadísticas que se va a analizar es el número de figuras. En un primer intento se hizo un conteo de las imágenes del documento pero se vio que no cuadraban. Las razones son diversas, entre ellas: las figuras pueden componerse de varias imágenes, los artículos pueden tener una breve biografía de los autores con su fotografía, las cabeceras pueden incluir imágenes de la publicación, etc.

Aunque queda demostrado que no tiene utilidad para contabilizar las figuras, se ha man-

tenido esta estadística para su relación con el número de figuras.

Para realizar el conteo, se buscan las imágenes del documento y se anota su identificador. Este identificador no se especifica en la documentación de la librería PyPDF2. Se obtuvo la información de cómo se almacenan las imágenes revisando el código en el repositorio público.

Los identificadores de objeto no se encuentran en todas las imágenes. Cuando es nulo, se toma como identificador el nombre de la imagen.

Al finalizar la inspección del documento se cuentan todos los identificadores únicos, sin repeticiones, para obtener el número total de imágenes.

Número de figuras

El número de figuras se calcula de un modo similar al que se emplea para el número de tablas. Como ocurría en el caso anterior, no se consiguió mediante el uso de la librería `pdfminer.six`.

En el apartado anterior se describe el primer método empleado, que es contabilizando imágenes. Tras abandonar este método se decide emplear el mismo que las tablas, usando expresiones regulares.

Se trata de identificar un bloque de texto que comience por el texto «figure» o simplemente su abreviatura «fig». Luego se busca el número identificativo que le sigue y se anota para descartar los valores repetidos. Al finalizar la lectura de todo el documento se obtiene una lista de valores únicos de figuras.

4.5. Presentación

La salida de datos se realiza en XML. Para las etiquetas de los metadatos se sigue el estándar Dublin Core. Aparte se añaden algunas propiedades personalizadas para los metadatos de conteo: número de páginas, de tablas y de figuras. Estos se incluyen en forma de atributos, junto con el nombre del fichero.

Ejemplo del código XML generado con los metadatos de un artículo de investigación:

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
   >
3   <paper filename="paper.pdf" pages="19" tables="5"
```

figures="9">

4 <dc:creator>Yuan Li</dc:creator>
5 <dc:creator>Mayire Ibrayim</dc:creator>
6 <dc:creator>Askar Hamdulla</dc:creator>
7 <dc:description>In the last years, methods for detecting text in real scenes have made significant progress with an increase in neural networks. However, due to the limitation of the receptive field of the central nervous system and the simple representation of text by using rectangular bounding boxes, the previous methods may be insufficient for working with more challenging instances of text. To solve this problem, this paper proposes a scene text detection network based on cross-scale feature fusion (CSFF-Net). The framework is based on the lightweight backbone network Resnet, and the feature learning is enhanced by embedding the depth weighted convolution module (DWCM) while retaining the original feature information extracted by CNN. At the same time, the 3D-Attention module is also introduced to merge the context information of adjacent areas, so as to refine the features in each spatial size. In addition, because the Feature Pyramid Network (FPN) cannot completely solve the interdependence problem by simple element-wise addition to process crosslayer information flow, this paper introduces a Cross-Level Feature Fusion Module (CLFFM) based on FPN, which is called Cross-Level Feature Pyramid Network (Cross-Level FPN). The

```

proposed CLFFM can better handle cross-layer
information flow and output detailed feature
information, thus improving the accuracy of text
region detection. Compared to the original network
framework, the framework provides a more advanced
performance in detecting text images of complex
scenes, and extensive experiments on three
challenging datasets validate the realizability of
our approach.</dc:description>
8   <dc:format>application/pdf</dc:format>
9   <dc:language>en</dc:language>
10  <dc:publisher>LaTeX with hyperref</dc:publisher>
11  <dc:subject>feature extraction</dc:subject>
12  <dc:subject>attention mechanism</dc:subject>
13  <dc:subject>pyramid network</dc:subject>
14  <dc:subject>deep learning</dc:subject>
15  <dc:subject>text detection</dc:subject>
16  <dc:title>Article CSFF-Net: Scene Text Detection
    Based on Cross-Scale Feature Fusion</dc:title>
17  </paper>
18 </metadata>

```

4.6. Pruebas

Se han realizado una serie de pruebas de las clases creadas. Como caso ilustrativo se mostrará el del metadato título de la clase PdfReader.

```

1  def test_abstract(self):
2      for item in self.pa:
3          self.assertEqual(item['reader'].abstract(),
4                           item['tsv']['description'])

```

```

5  def test_figure_count(self):
6      for item in self.pa:
7          self.assertEqual(item['reader'].
            figure_count, int(item['tsv']['figures']))
8
9  def test_keywords(self):
10     for item in self.pa:
11         subject = item['tsv']['subject'].split(' ;
            ') if item['tsv']['subject'] else []
12         self.assertEqual(item['reader'].keywords(),
            subject)
13
14 def test_table_count(self):
15     for item in self.pa:
16         self.assertEqual(item['reader'].table_count
            , int(item['tsv']['tables']))
17
18 def test_title(self):
19     for item in self.pa:
20         self.assertEqual(item['reader'].title(),
            item['tsv']['title'])

```

```
(base) jamd@localhost:/media/TFG/Jupyter/test> python3 test_pdf_reader.py
test_abstract (__main__.TestPdfReader) ... ok
test_figure_count (__main__.TestPdfReader) ... ok
test_keywords (__main__.TestPdfReader) ... ok
test_table_count (__main__.TestPdfReader) ... ok
test_title (__main__.TestPdfReader) ... ok

-----
Ran 5 tests in 5.820s

OK
(base) jamd@localhost:/media/TFG/Jupyter/test> █
```

Figura 14: Resultados del test de la clase PdfReader

5

Resultados

A continuación se muestran ejemplos de aplicar el *software* desarrollado en los artículos de muestra.

En primer lugar se ha calculado cuántos metadatos tienen introducidos los PDF. Solo se realiza un conteo de los cien artículos de muestra. Los metadatos contenidos en DocumentInfo tienen el prefijo «doc_». Los metadatos de XMPInformation tienen el prefijo «xmp_» (Figura 15).

Tras el proceso de análisis y extracción de metainformación, el resultado de los metadatos no vacíos se muestra en la figura 16.

En las figuras 17, 18 y 19 se muestran las salidas de un cuaderno Jupyter para un artículo.

Por último, en la figura 20 se ve la salida generada en XML con los datos extraídos de todos los artículos del directorio.

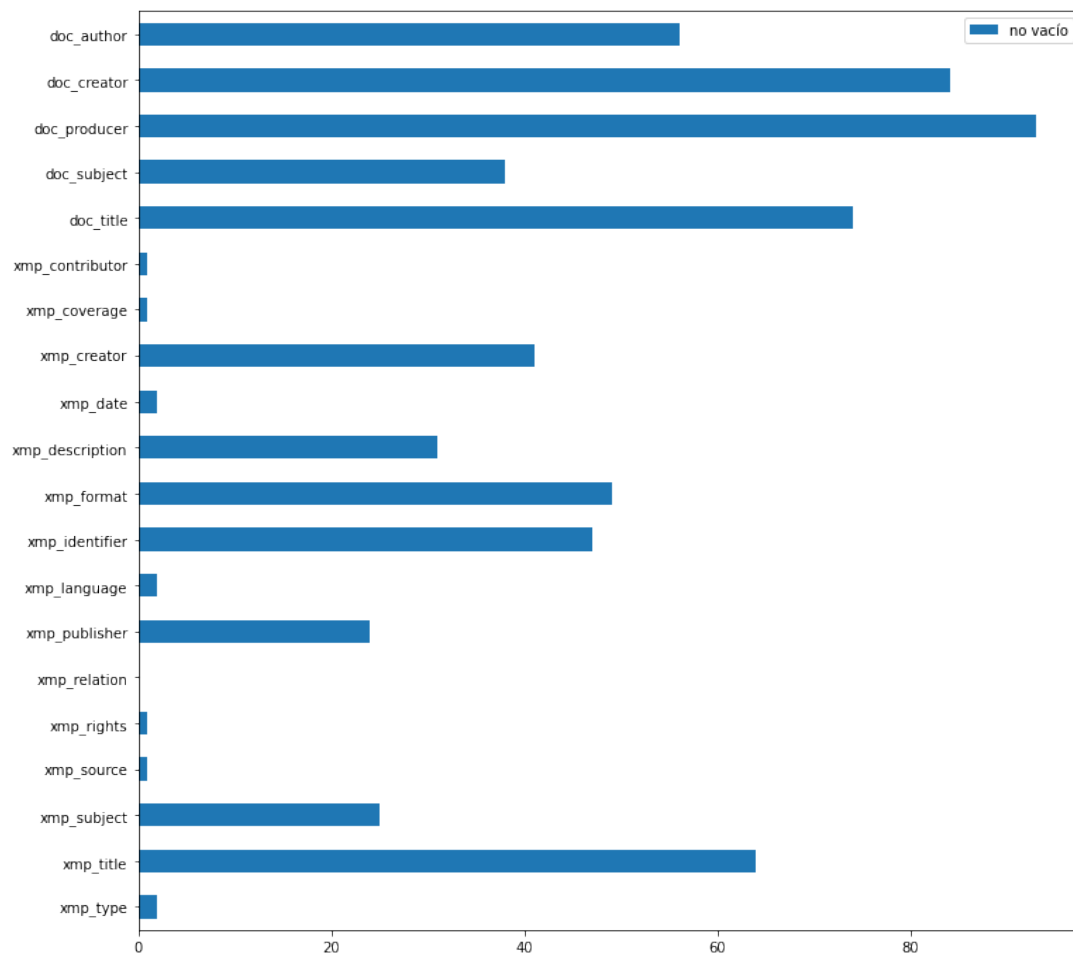


Figura 15: Conteo de metadatos incluidos en los PDF

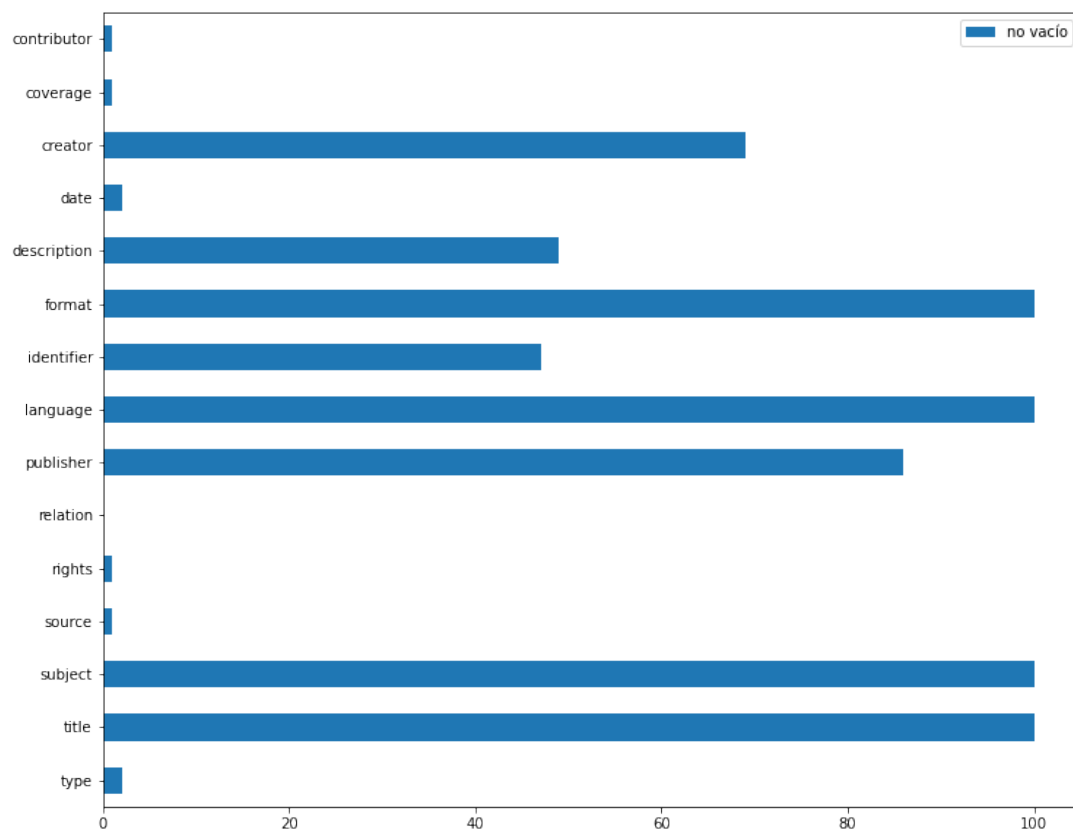


Figura 16: Conteo de metadatos tras analizar los artículos

```
In [10]: pa.stats_dataframe()
```

Out[10]:

paper-01.pdf	
pages	19
figures	9
tables	5
images	14

Figura 17: Estadísticas de un artículo en Jupyter


```
In [11]: df = pa.dc_dataframe()
df = df.style.set_properties(**{'text-align': 'left'})
df.set_table_styles([dict(selector='th', props=[('text-align', 'center')])]
df
```

paper-01.pdf	
contributor	[]
coverage	[]
creator	['Yuan LI', 'Mayire Ibrayim', 'Askar Hamdulla']
date	[]
description	['In the last years, methods for detecting text in real scenes have made significant progress with an increase in neural networks. However, due to the limitation of the receptive field of the central nervous system and the simple representation of text by using rectangular bounding boxes, the previous methods may be insufficient for working with more challenging instances of text. To solve this problem, this paper proposes a scene text detection network based on cross-scale feature fusion (CSFF-Net). The framework is based on the lightweight backbone network Resnet, and the feature learning is enhanced by embedding the depth weighted convolution module (DWCM) while retaining the original feature information extracted by CNN. At the same time, the 3D-Attention module is also introduced to merge the context information of adjacent areas, so as to refine the features in each spatial size. In addition, because the Feature Pyramid Network (FPN) cannot completely solve the interdependence problem by simple element-wise addition to process crosslayer information flow, this paper introduces a Cross-Level Feature Fusion Module (CLFFM) based on FPN, which is called Cross-Level Feature Pyramid Network (Cross-Level FPN). The proposed CLFFM can better handle cross-layer information flow and output detailed feature information, thus improving the accuracy of text region detection. Compared to the original network framework, the framework provides a more advanced performance in detecting text images of complex scenes, and extensive experiments on three challenging datasets validate the realizability of our approach.']
format	['application/pdf']
identifier	[]
language	['en']
publisher	['LaTeX with hyperref']
relation	[]
rights	[]
source	[]
subject	['feature extraction', 'attention mechanism', 'pyramid network', 'deep learning', 'text detection']
title	['Article CSFF-Net: Scene Text Detection Based on Cross-Scale Feature Fusion']
type	[]

Figura 18: Metadatos de un artículo en Jupyter

```
In [12]: from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = pa._reader.text()
wordcloud = WordCloud(max_font_size=100, max_words=25, background_color="white").generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Figura 19: Nube de palabras de un artículo en Jupyter

```

In [4]: import IPython

IPython.display.Code(pf.to_xml(), language='xml')

Out[4]: <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
  <paper filename="paper-01.pdf" pages="19" tables="5" figures="9">
    <dc:creator>Yuan Li</dc:creator>
    <dc:creator>Mayire Ibrayim</dc:creator>
    <dc:creator>Askar Hamdulla</dc:creator>
    <dc:description>In the last years, methods for detecting text in real scenes have made significant progress with an increase in neural networks. However, due to the limitation of the receptive field of the central nervous system and the simple representation of text by using rectangular bounding boxes, the previous methods may be insufficient for working with more challenging instances of text. To solve this problem, this paper proposes a scene text detection network based on cross-scale feature fusion (CSFF-Net). The framework is based on the lightweight backbone network Resnet, and the feature learning is enhanced by embedding the depth weighted convolution module (DWCN) while retaining the original feature information extracted by CNN. At the same time, the 3D-Attention module is also introduced to merge the context information of adjacent areas, so as to refine the features in each spatial size. In addition, because the Feature Pyramid Network (FPN) cannot completely solve the interdependence problem by simple element-wise addition to process crosslayer information flow, this paper introduces a Cross-Level Feature Fusion Module (CLFFM) based on FPN, which is called Cross-Level Feature Pyramid Network (Cross-Level FPN). The proposed CLFFM can better handle cross-layer information flow and output detailed feature information, thus improving the accuracy of text region detection. Compared to the original network framework, the framework provides a more advanced performance in detecting text images of complex scenes, and extensive experiments on three challenging datasets validate the realizability of our approach.</dc:description>
    <dc:format>application/pdf</dc:format>
    <dc:language>en</dc:language>
    <dc:publisher>LaTeX with hyperref</dc:publisher>
    <dc:subject>feature extraction</dc:subject>
    <dc:subject>attention mechanism</dc:subject>
    <dc:subject>pyramid network</dc:subject>
    <dc:subject>deep learning</dc:subject>
    <dc:subject>text detection</dc:subject>
    <dc:title>Article CSFF-Net: Scene Text Detection Based on Cross-Scale Feature Fusion</dc:title>
  </paper>
  <paper filename="paper-02.pdf" pages="13" tables="0" figures="15">
    <dc:description>Winding function (WF) method is one of the electric machines modelling methods, which is used to simulate electric machines under different conditions. It has less simulation time and memory than other methods and, at the same time, has comparable accuracy over other methods. One of the significant challenges of WF-based models is how to model saturation. The accuracy of the fault detection method in electric machines depends on the accuracy of the model and the method used for the simulation. A new extended model of the WF is presented for saturation modelling, which considers all parts of the core including teeth, slots, and yokes. In this method, the effect of the iron parts is emulated as a partial value that is added to the length of the actual air gap. The simulation results for normal and faulty conditions are shown in two cases, with and without considering saturation effect. Then the importance of saturation modelling has been demonstrated. Simulation is done in MATLAB software. Moreover, the experimental results are shown to verify the proposed model.</dc:description>
    <dc:format>application/pdf</dc:format>
    <dc:language>en</dc:language>
    <dc:publisher>Arbortext Advanced Print Publisher 11.2.5182/W-x64</dc:publisher>
    <dc:subject>saturation effect</dc:subject>
    <dc:subject>flux density</dc:subject>
    <dc:subject>etal</dc:subject>
    <dc:subject>torque</dc:subject>
    <dc:subject>air gap</dc:subject>
    <dc:title>An extended winding function model for induction machine modelling considering saturation effect</dc:title>
  </paper>
</metadata>

```

Figura 20: XML del directorio de artículos en Jupyter

6

Conclusiones

El trabajo con librerías dedicadas al manejo de PDF. Se ha descubierto que no todas manejan los documentos de igual modo. Esto me ha servido para poder trabajar con las dos seleccionadas para obtener la información que requería.

Los metadatos no siempre se rellenan acorde a lo esperado. Examinando las muestras se ha visto que no se suelen rellenar todos los campos. Y, en algunos casos, se emplean las etiquetas equivocadas.

Los artículos siguen un esquema muy similar, salvo excepciones. Esto facilita la extracción de información. Se han podido conseguir resultados desde el primer momento gracias a esto.

El procesamiento de lenguaje natural ha resultado muy útil para obtener términos frecuentes. Gracias al proyecto, he conocido cómo se preparan los textos y se trabaja con la librería `nltk` de Python.

Otro aspecto fundamental, es ver la importancia de usar un esquema como Dublin Core. El hecho de usar un sistema estandarizado de metainformación es una ventaja para trasladar estos datos entre sistemas.

6.1. Líneas futuras

Este trabajo podría complementarse por dos vías: rendimiento y eficiencia.

El rendimiento se puede mejorar utilizando multiproceso. Los diferentes cálculos que se realizan con cada PDF consumen mucho tiempo. Habría que estudiar que métodos serían convenientes para lanzar varios análisis de artículos a la vez.

La eficiencia se puede mejorar utilizando técnicas más nuevas dentro de la inteligencia artificial. El año 2017, con la publicación del artículo «Attention Is All You Need», trajo un nuevo modelo de redes neuronales, los Transformer. Estos se caracterizan por recordar todo el texto introducido, a diferencia de los modelos recursivos. Habría que estudiar si este nuevo

modelo mejora los resultados obtenidos y compensa el coste de cálculo que supone usar una red neuronal.

Referencias

- [1] W. To y B. T. Yu, “Rise in higher education researchers and academic publications,” *Emerald Open Research*, vol. 2, pág. 3, 2020.
- [2] “Anaconda Distribution,” Anaconda. (2017), dirección: <https://blog.desdelinux.net/wp-content/uploads/2017/09/Anaconda-Distribution-Diagram.png.webp>.
- [3] “Anaconda | The World’s Most Popular Data Science Platform,” Anaconda. (2022), dirección: <https://www.anaconda.com>.
- [4] S. Seth, S. Rüping y S. Wrobel, “Metadata Extraction using Text Mining,” en *HealthGrid*, 2009, págs. 95-104.
- [5] S. S. Bhanuse, S. D. Kamble y S. M. Kakde, “Text mining using metadata for generation of side information,” *Procedia Computer Science*, vol. 78, págs. 807-814, 2016.
- [6] C. Zong, R. Xia y J. Zhang, *Text Data Mining*. Springer, 2021, vol. 711.
- [7] “Dublin Core specification,” Dublin Core Metadata Initiative. (2022), dirección: <https://www.dublincore.org/specifications/dublin-core/> (visitado 2022).
- [8] E. Méndez, “Dublin Core, metadatos y vocabularios,” *El profesional de la información*, 2006.
- [9] “XML and JSON Data Modeling Best Practices,” MarkLogic University. (2019), dirección: <https://www.youtube.com/watch?v=6t2dYJjM4tg>.
- [10] “Document management — Portable document format — Part 1: PDF 1.7,” Adobe Systems Incorporated. (2008), dirección: https://opensource.adobe.com/dc-acrobat-sdk-docs/pdfstandards/PDF32000_2008.pdf.
- [11] M. Fenniak, *PyPDF2 documentation*, 2008. dirección: <https://pypdf2.readthedocs.io/en/latest/>.
- [12] Y. Shinyama, P. Guglielmetti y P. Marsman, *pdfminer.six documentation*, 2019. dirección: <https://pdfminersix.readthedocs.io/en/latest/>.

Apéndice A

Manual de instalación

Para que se pueda ejecutar el código y visualizar los resultados se requiere la distribución Anaconda. Además, es necesario instalar las librerías de Python: PyPdf2, pdfminer.six, langdetect y wordcloud.

El sistema operativo que se ha empleado es la distribución de Linux OpenSUSE Tumbleweed. Los pasos que se muestran en este manual están basados en dicha distribución.

Requisitos:

- Licencia: uso y distribución libre bajo los términos del Acuerdo de Licencia de Usuario Final de Anaconda Distribution.
- Sistema operativo: Windows 8 o posterior, Linux o macOS 10.13+.
- Arquitectura de sistema: 64-bit x86 y 32-bit x86 (solo Windows)
- Espacio en disco: mínimo 5 GB para la descarga e instalación.

Instalación en Linux:

Prerrequisitos

Para utilizar los paquetes de la interfaz gráfica con Linux, se deben instalar las siguientes dependencias con Qt:

- libXcomposite1
- libXi6
- libXext6
- libXau6
- libX11-6
- libXrandr2
- libXrender1
- libXss1
- libXtst6

- libXdamage1
- libXcursor1
- libxcb1
- libasound2
- libX11-xcb1
- Mesa-libGL1
- Mesa-libEGL1

Se puede comprobar si se encuentran instalados con el comando:

```
zypper search libXcomposite1 libXi6 libXext6 libXau6 libX11-6
libXrandr2 libXrender1 libXss1 libXtst6 libXdamage1
libXcursor1 libxcb1 libasound2 libX11-xcb1 Mesa-libGL1 Mesa
-libEGL1
```

Si no se encontraran en el sistema, se instalan con el siguiente comando:

```
zypper install libXcomposite1 libXi6 libXext6 libXau6 libX11
-6 libXrandr2 libXrender1 libXss1 libXtst6 libXdamage1
libXcursor1 libxcb1 libasound2 libX11-xcb1 Mesa-libGL1 Mesa
-libEGL1
```

Descarga

Desde el navegador, se accede a la web de Anaconda y se selecciona el instalador de Linux.

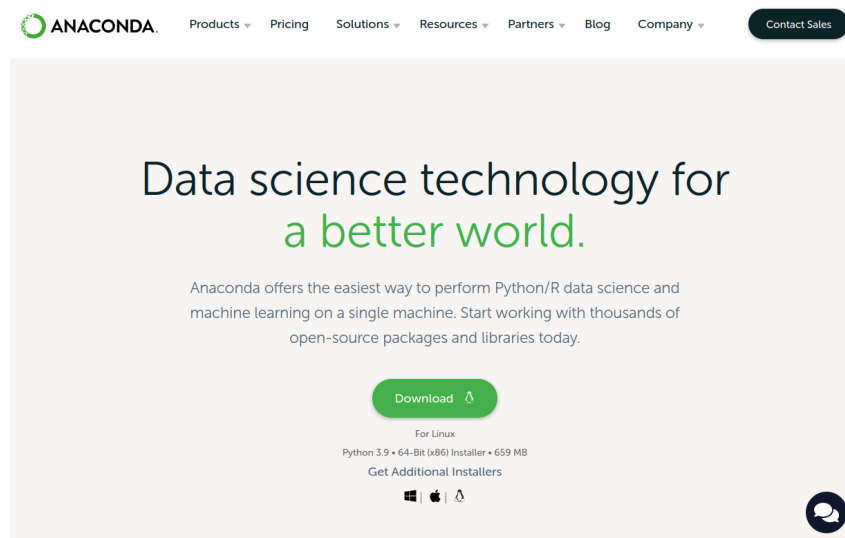


Figura 21: Página de inicio de la web de Anaconda [3]

Una vez descargado el fichero, se recomienda verificar la integridad del mismo. Desde el terminal se ejecuta el siguiente comando:

```
shasum -a 256 Anaconda3-2022.05-Linux-x86_64.sh
```

El resultado debe coincidir con el *hash* facilitado por Anaconda en su web <https://docs.anaconda.com/anaconda/install/hashes/>.


Instalación

Desde la terminal hay que situarse en la carpeta de descargas. Luego hay que lanzar el ejecutable con Bash:

```
bash ./Anaconda3-2020.05-Linux-x86_64.sh
```

Se solicita la revisión del acuerdo de licencia. Hay que pulsar Enter para continuar. Una vez leído, se introduce «yes» si se aceptan los términos y se quiere continuar con la instalación.

A continuación se solicita la ruta de instalación. Se recomienda utilizar la ruta por defecto (Figura 22).



```
- Press ENTER to confirm the location  
- Press CTRL-C to abort the installation  
- Or specify a different location below
```

Figura 22: Mensaje de instalación de Anaconda

Una vez finalizada la instalación, hay que elegir si se inicializa Anaconda Distribution. Se recomienda aceptar la ejecución de `conda init`. Si no se aceptara, no se modificará el *shell* y habrá que activar conda cada vez que se quiera lanzar.

Verificación

Tras la instalación, se puede verificar que todos los paquetes están instalados en el sistema ejecutando desde la terminal el comando:

```
conda list
```

Por pantalla deben aparecer todos los paquetes de conda junto con su versión, como se muestra en la Figura 23.

Por último, también desde la terminal, se ejecuta el comando:

```
anaconda-navigator
```

#	Name	Version	Build	Channel
	_ipyw_jlab_nb_ext_conf	0.1.0	py39h06a4308_1	
	_libgcc_mutex	0.1	main	
	_openmp_mutex	4.5	1_gnu	
	aiohttp	3.8.1	py39h7f8727e_1	
	aiosignal	1.2.0	pyhd3eb1b0_0	
	alabaster	0.7.12	pyhd3eb1b0_0	
	anaconda	2022.05	py39_0	
	anaconda-client	1.9.0	py39h06a4308_0	
	anaconda-navigator	2.1.4	py39h06a4308_0	
	anaconda-project	0.10.2	pyhd3eb1b0_0	
	anyio	3.5.0	py39h06a4308_0	
	appdirs	1.4.4	pyhd3eb1b0_0	
	argon2-cffi	21.3.0	pyhd3eb1b0_0	
	argon2-cffi-bindings	21.2.0	py39h7f8727e_0	
	arrow	1.2.2	pyhd3eb1b0_0	
	astroid	2.6.6	py39h06a4308_0	
	astropy	5.0.4	py39hce1f21e_0	
	asttokens	2.0.5	pyhd3eb1b0_0	
	async-timeout	4.0.1	pyhd3eb1b0_0	
	atomicwrites	1.4.0	py_0	
	attrs	21.4.0	pyhd3eb1b0_0	
	automat	20.2.0	py_0	
	autopep8	1.6.0	pyhd3eb1b0_0	
	babel	2.9.1	pyhd3eb1b0_0	
	backcall	0.2.0	pyhd3eb1b0_0	
	backports	1.1	pyhd3eb1b0_0	
	backports.functools_lru_cache	1.6.4	pyhd3eb1b0_0	
	backports.tempfile	1.0	pyhd3eb1b0_1	
	backports weakref	1.0.post1	py_1	
	bcrypt	3.2.0	py39he8ac12f_0	
	beautifulsoup4	4.11.1	py39h06a4308_0	
	binaryornot	0.4.4	pyhd3eb1b0_1	
	bitarray	2.4.1	py39h7f8727e_0	
	bkcharts	0.2	py39h06a4308_0	
	black	19.10b0	py_0	
	blas	1.0	mkl	
	bleach	4.1.0	pyhd3eb1b0_0	
	blosc	1.21.0	h8c45485_0	
	bokeh	2.4.2	py39h06a4308_0	
	boto3	1.21.32	pyhd3eb1b0_0	
	botocore	1.24.32	pyhd3eb1b0_0	
	bottleneck	1.3.4	py39hce1f21e_0	
	brtli	1.0.9	he6710b0_2	
	brtliipy	0.7.0	py39h27cfd23_1003	
	brunsl	0.1	h2531618_0	
	bzip2	1.0.8	h7b6447c_0	
	c-ares	1.18.1	h7f8727e_0	
	ca-certificates	2022.3.29	h06a4308_1	
	cachetools	4.2.2	pyhd3eb1b0_0	

Figura 23: Listado de paquetes instalados

Este abrirá la ventana de Anaconda Navigator. Esta aplicación sirve de lanzador de todo el software que incluye Anaconda (Figura 24).

Librería adicionales

Además de todos los paquetes que incluye conda, es necesario instalar cuatro librerías adicionales. Estas son utilizadas para la lectura de los PDF, la detección del idioma y la generación de nubes de palabras.

La instalación se realiza desde la terminal. Hay dos métodos para hacerlo:

- El repositorio Python Package Index (PyPI). Se utiliza el comando pip.

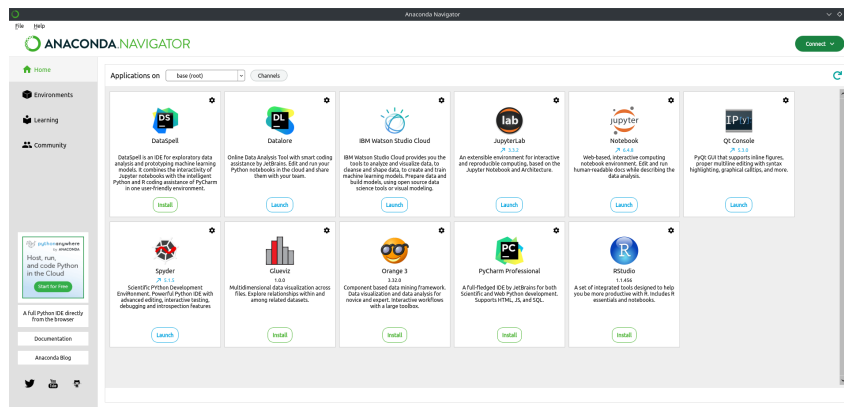


Figura 24: Captura de la pantalla de inicio de Anaconda Navigator

- La colección de software conda-forge mantenido por la comunidad. Se utiliza el comando conda.

Instalación mediante PyPI:

```
pip install langdetect
pip install pdfminer.six
pip install PyPDF2
pip install wordcloud
```

Instalación mediante conda-forge:

```
conda install -c conda-forge langdetect
conda install -c conda-forge pdfminer.six
conda install -c conda-forge pypdf2
conda install -c conda-forge wordcloud
```



UNIVERSIDAD
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga