Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia artificial

Sistema OCR multimotor basado en IA para PDFs escaneados en español

Trabajo fin de estudio presentado por:	Sergio Jiménez Jiménez
Tipo de trabajo:	Desarrollo Software
Director/a:	Javier Rodrigo Villazón Terrazas
Fecha:	06.10.2025

Resumen

Este Trabajo Fin de Máster presenta el desarrollo de un sistema OCR multimotor basado en aprendizaje profundo para la extracción de texto desde documentos PDF escaneados en español.

El objetivo principal es evaluar y comparar diferentes motores OCR neuronales EasyOCR, PaddleOCR y DocTR, con el fin de determinar cuál ofrece el mejor equilibrio entre precisión (CER/WER) y eficiencia en entornos reales.

El proyecto implementa un pipeline modular que automatiza las fases de conversión de PDF a imagen, preprocesado, reconocimiento óptico y evaluación. El sistema está diseñado con herramientas open source en Python, integrando librerías como PyMuPDF, OpenCV y pandas, lo que garantiza su reproducibilidad y portabilidad.

Los resultados preliminares muestran que EasyOCR ofrece el mejor equilibrio global entre precisión y velocidad, mientras que PaddleOCR logra detecciones más robustas en documentos complejos y DocTR destaca en el reconocimiento a nivel de carácter.

Estos hallazgos demuestran la viabilidad técnica del sistema y sientan las bases para una optimización futura de arquitecturas OCR neuronales en español.

Abstract

This master's Thesis presents the development of a multi-engine OCR system based on deep learning for text extraction from scanned PDF documents in Spanish.

The main goal is to evaluate and compare modern neural OCR engines EasyOCR, PaddleOCR, and DocTR, to identify the best trade-off between accuracy (CER/WER) and efficiency in real-world scenarios.

The proposed system implements a modular pipeline automating all stages: PDF-to-image conversion, image preprocessing, optical character recognition, and quantitative evaluation. It is developed in Python using open-source libraries such as PyMuPDF, OpenCV, and pandas, ensuring full reproducibility and transparency.

Preliminary results show that EasyOCR provides the best overall balance between precision and performance, PaddleOCR delivers robust text detection, and DocTR achieves high character-level accuracy.

These findings confirm the system's technical feasibility and establish a solid foundation for future research on neural OCR architectures optimized for the Spanish language.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

En los últimos años, la digitalización masiva de archivos ha transformado la manera en que instituciones públicas, empresas y universidades gestionan la información. Sin embargo, una gran parte de los documentos conservados en formato digital, especialmente en PDFs escaneados contienen únicamente imágenes sin capa de texto reconocible. Esto impide su búsqueda, indexación y análisis automático, reduciendo drásticamente su utilidad práctica en entornos administrativos, educativos y científicos.

El reconocimiento óptico de caracteres (OCR) busca resolver este problema transformando imágenes de texto en texto editable. No obstante, los motores OCR clásicos, como Tesseract o ABBYY FineReader, presentan un rendimiento insuficiente cuando los documentos incluyen ruido, inclinaciones, artefactos de compresión o tipografías diversas, y su precisión disminuye aún más en idiomas con tildes o caracteres especiales, como el español.

Diversas investigaciones recientes han demostrado que las arquitecturas basadas en aprendizaje profundo, particularmente aquellas que combinan redes convolucionales (CNN) y modelos secuenciales o Transformer, superan en precisión y robustez a las soluciones tradicionales. Ejemplos de estos avances incluyen PaddleOCR (Baidu, 2020), EasyOCR (JaidedAI, 2020) o DocTR (Mindee, 2021), los cuales integran en un único pipeline la detección y el reconocimiento de texto.

La relevancia de este trabajo radica en aplicar, evaluar y comparar estos motores neuronales en el contexto específico del idioma español, donde la investigación y las métricas de rendimiento son todavía escasas. Desde una perspectiva científica y práctica, el proyecto aporta una herramienta de análisis que facilita la extracción de texto de calidad desde PDFs escaneados, promoviendo la accesibilidad y el aprovechamiento de información textual en el ámbito académico y documental.

En este sentido, la motivación principal es contribuir al desarrollo de soluciones inteligentes, reproducibles y orientadas a datos reales, que reduzcan la brecha entre las capacidades del OCR tradicional y el potencial del reconocimiento basado en inteligencia artificial.

1.2. Planteamiento del trabajo

El problema de partida identificado es la dificultad para extraer texto fiable desde documentos PDF escaneados, especialmente cuando estos presentan baja calidad, inclinaciones, ruido o diferentes formatos de maquetación. En estos casos, los motores OCR tradicionales no alcanzan la precisión necesaria para su aplicación en entornos productivos, administrativos o académicos, donde la búsqueda y gestión de información textual es una tarea clave.

Para dar respuesta a esta necesidad, se plantea el desarrollo de un sistema OCR multimotor basado en inteligencia artificial, capaz de integrar diferentes modelos de reconocimiento neuronal en un mismo entorno de ejecución. Esta propuesta permite comparar, validar y optimizar el proceso de reconocimiento óptico de caracteres aplicando principios de la ingeniería de software, como modularidad, mantenibilidad y reutilización de componentes.

La finalidad del trabajo es construir un pipeline reproducible y escalable, desarrollado en Python, que abarque todas las fases del proceso de extracción: desde la conversión de PDF a imagen, el preprocesado de las páginas mediante técnicas de visión por computador (OpenCV), la ejecución del reconocimiento con motores OCR neuronales (EasyOCR, PaddleOCR y DocTR) y la evaluación automática de los resultados mediante métricas cuantitativas (CER y WER).

Este enfoque combina percepción computacional y aprendizaje automático dentro del marco de la ingeniería de software, contribuyendo al desarrollo de soluciones inteligentes de análisis documental.

Además, el sistema se concibe como una herramienta abierta y extensible, capaz de incorporar en el futuro nuevos modelos OCR o módulos de mejora lingüística, fomentando la investigación aplicada y la transferencia tecnológica en el ámbito del procesamiento de documentos digitales.

1.3. Estructura del trabajo

El presente documento se estructura en cinco capítulos que permiten seguir de forma ordenada el proceso de desarrollo y evaluación del sistema OCR multimotor propuesto.

Capítulo 1 Introducción: presenta el contexto general del problema, la motivación y los objetivos que justifican el desarrollo del proyecto. También introduce brevemente la metodología y los resultados esperados.

Capítulo 2 Contexto y estado del arte: analiza el marco teórico y tecnológico del reconocimiento óptico de caracteres (OCR), revisando las principales soluciones existentes basadas en aprendizaje profundo, así como los trabajos previos más relevantes.

Capítulo 3 Objetivos concretos y metodología de trabajo: detalla los objetivos específicos, el enfoque metodológico adoptado y las métricas de evaluación que se emplearán para validar los resultados del sistema.

Capítulo 4 Desarrollo del sistema: describe la implementación del pipeline OCR, los módulos de software, las herramientas empleadas y la integración de los motores EasyOCR, PaddleOCR y DocTR.

Capítulo 5 Resultados y conclusiones: presenta los resultados obtenidos, analiza su significado y propone líneas de mejora y continuidad para trabajos futuros.

2. Contexto y estado del arte

El reconocimiento óptico de caracteres (OCR) constituye una de las áreas más consolidadas dentro del campo de la visión por computador aplicada al procesamiento de documentos. Su finalidad es convertir imágenes de texto procedentes de escáneres, cámaras o archivos PDF rasterizados en contenido digital editable y analizable.

A pesar de su antigüedad, el OCR sigue enfrentando retos considerables. Los documentos reales suelen presentar ruido, distorsiones, diferentes resoluciones, inclinaciones o variaciones tipográficas, lo que dificulta la segmentación y el reconocimiento preciso del texto. Estas limitaciones afectan especialmente a documentos en idiomas no dominantes, como el español, que incluye acentos, eñes y signos diacríticos poco representados en datasets globales.

En entornos administrativos, jurídicos y académicos, la extracción automática de texto de documentos escaneados tiene una relevancia directa para la automatización de flujos documentales, digitalización de archivos históricos y accesibilidad a la información. Por ello, se considera una línea de investigación y desarrollo prioritaria dentro de la ingeniería de software orientada a la gestión documental y la inteligencia artificial aplicada.

2.1.Contexto del problema

El reconocimiento óptico de caracteres (OCR) es una de las tecnologías más extendidas dentro del campo de la visión artificial y el procesamiento de documentos digitales. Su objetivo es convertir texto impreso presente en imágenes o documentos escaneados en texto editable y analizable, facilitando su almacenamiento, búsqueda y explotación automatizada.

En los entornos donde el volumen de documentación física es elevado, como la administración pública, el ámbito jurídico o las instituciones educativas, la conversión de archivos escaneados a texto resulta fundamental para avanzar hacia la automatización de procesos y la gestión documental inteligente.

A pesar de los progresos en digitalización, muchos archivos PDF contienen únicamente imágenes rasterizadas, sin capa de texto embebida. Este tipo de documentos imposibilita su lectura automática mediante buscadores o herramientas de análisis, lo que limita su valor informativo.

Los sistemas OCR clásicos, basados en métodos heurísticos y plantillas, tienen un rendimiento aceptable solo en condiciones óptimas (texto limpio, tipografía homogénea, alineación correcta). Sin embargo, fallan ante factores comunes en escaneos reales:

- Baja resolución o compresión.
- Presencia de ruido o sombras.
- Inclinaciones (skew)
- Variaciones tipográficas o formatos de documento mixtos.

Estos problemas son especialmente relevantes en documentos en español, un idioma con signos diacríticos (tildes, eñes, interrogaciones iniciales) que suelen generar errores en los modelos entrenados con datasets anglosajones.

El avance de las técnicas de aprendizaje profundo (Deep Learning) ha supuesto un cambio de paradigma, permitiendo la aparición de sistemas OCR neuronales que combinan redes convolucionales (CNNs) para la detección de texto con modelos secuenciales (RNNs o Transformers) para su reconocimiento. Esta nueva generación de motores ha demostrado un rendimiento superior en entornos reales, gracias a su capacidad para aprender representaciones robustas a variaciones de imagen y ruido.

2.2. Estado del arte

En los últimos años, diferentes grupos de investigación y empresas tecnológicas han desarrollado soluciones OCR basadas en arquitecturas neuronales end-to-end, donde la detección y el reconocimiento se realizan de forma conjunta. A continuación, se resumen las más relevantes y su relación con este trabajo.

EasyOCR (Jaided AI, 2020)

Basado en DBNet para detección y CRNN o Transformer para reconocimiento, EasyOCR ofrece una solución ligera y de fácil implementación en Python. Su principal ventaja es la compatibilidad con más de 80 idiomas, incluido el español, y su ejecución sin necesidad de configuración compleja. Es ideal como línea base neuronal por su equilibrio entre precisión y velocidad.

PaddleOCR (Baidu, 2020)

Pertenece al ecosistema PaddlePaddle AI, desarrollado por Baidu. Implementa el modelo PP-OCR, que integra tres componentes: detector de texto (DB), clasificador de orientación (CLS) y reconocedor (SRN o CRNN). Se caracteriza por su robustez industrial, soporte multilingüe y documentación exhaustiva. Además, proporciona modelos optimizados para CPU y GPU, siendo uno de los referentes en OCR aplicado a producción.

DocTR (Mindee, 2021)

Es una librería open source en PyTorch y TensorFlow, orientada a la investigación académica. Utiliza arquitecturas como LinkNet o DBNet para la detección, y modelos CRNN, SAR o ViTSTR para el reconocimiento. Ofrece una salida estructurada que permite identificar texto a nivel de bloques, líneas y palabras, lo que facilita el análisis documental avanzado.

TrOCR (Microsoft, 2021)

Desarrollado por Microsoft Research, TrOCR (Transformer-based OCR) utiliza una arquitectura Vision Encoder–Decoder inspirada en modelos de traducción automática. Si bien alcanza alta precisión en el reconocimiento de texto, no incluye módulo de detección, por lo que no se utiliza directamente en este trabajo, aunque representa una línea futura de integración como reconocedor especializado.

2.3.Conclusiones

El estudio del dominio demuestra que la tecnología OCR ha evolucionado desde soluciones basadas en reglas y plantillas hacia modelos neuronales end-to-end, capaces de aprender patrones de texto directamente desde imágenes. Estos avances han mejorado significativamente la precisión en entornos reales y la capacidad de generalización frente a variaciones visuales.

En el contexto del idioma español, la literatura y las implementaciones disponibles siguen siendo limitadas, especialmente en la evaluación comparativa entre diferentes motores OCR neuronales. Por ello, el presente trabajo aporta un valor añadido al proponer una plataforma experimental que permite evaluar, comparar y optimizar estos modelos bajo un mismo pipeline de procesamiento.

Los hallazgos del estado del arte orientan directamente el desarrollo del sistema:

- **EasyOCR** se emplea como modelo base por su ligereza y disponibilidad.
- **PaddleOCR** se integra como motor industrial de referencia.
- **DocTR** se adopta como solución investigativa modular, ideal para experimentar con nuevas arquitecturas.

3. Objetivos concretos y metodología de trabajo

3.1. Objetivo general

El objetivo general de este trabajo es desarrollar y validar un sistema OCR multimotor basado en inteligencia artificial capaz de mejorar en al menos un 15 % la precisión de reconocimiento de texto (medida mediante CER) respecto a un modelo neuronal base (EasyOCR), aplicado a documentos PDF escaneados en español, garantizando además un tiempo de procesamiento promedio por página inferior a tres segundos en entornos de hardware estándar (GPU RTX 3060 o equivalente).

Este objetivo cumple las características SMART:

- Específico: aborda un problema claramente definido; la extracción automática de texto de PDFs escaneados en español mediante modelos neuronales OCR.
- Medible: se cuantifica el éxito mediante las métricas CER (Character Error Rate), WER (Word Error Rate) y latencia media por página.
- Alcanzable: se basa en herramientas open source (Python, PyTorch, PaddleOCR, DocTR) y recursos de libre acceso, lo que hace viable su implementación dentro del tiempo y medios disponibles.
- Relevante: contribuye a la ingeniería de software aplicada al procesamiento documental y a la digitalización inteligente, líneas prioritarias del Máster.
- Temporal: se ha planificado su desarrollo y validación en un plazo de un semestre académico, siguiendo la estructura de entregables progresivos.

3.2. Objetivos específicos

Analizar las principales soluciones OCR basadas en aprendizaje profundo y seleccionar tres motores representativos (EasyOCR, PaddleOCR y DocTR) como base de comparación.

Desarrollar un pipeline modular en Python que implemente la conversión de PDF a imagen, el preprocesado con OpenCV y la integración de los modelos OCR neuronales seleccionados.

Evaluar el rendimiento de cada motor mediante métricas cuantitativas de precisión (CER y WER) y de eficiencia (latencia por página).

Comparar los resultados obtenidos para determinar el modelo con mejor equilibrio entre precisión y rendimiento en el contexto de documentos escaneados en español.

Documentar y validar el sistema a través de notebooks reproducibles y generar un informe técnico con análisis de resultados, limitaciones y propuestas de mejora.

3.3. Metodología del trabajo

El trabajo se ha estructurado siguiendo una metodología experimental y comparativa, centrada en el diseño y evaluación de software aplicado al reconocimiento óptico de caracteres.

Diseño general del sistema

El sistema propuesto se implementa en Python, utilizando librerías open source de procesamiento de imágenes y aprendizaje profundo. El flujo de trabajo general se representa de la siguiente forma:

PDF → Imagen (PyMuPDF/pdf2image) → Preprocesado (OpenCV) → OCR (EasyOCR | PaddleOCR | DocTR) → Evaluación (CER, WER, Latencia)

Herramientas y recursos

- Lenguaje y frameworks: Python 3.10, PyTorch, PaddlePaddle.
- OCR Engines: EasyOCR (v1.7+), PaddleOCR (v2.7+), DocTR (v0.7+).
- Procesamiento de imágenes: OpenCV, NumPy, pdf2image, PyMuPDF.
- Evaluación y análisis: pandas, rapidfuzz (distancia de Levenshtein).
- Hardware: CPU Intel i7, 32 GB RAM.
- Datasets: RVL-CDIP, FUNSD y documentos escaneados propios en españo

Métricas de evaluación

CER (Character Error Rate): proporción de caracteres erróneos respecto al texto de referencia.

WER (Word Error Rate): errores a nivel de palabra, considerando sustituciones, inserciones y omisiones.

Latencia media: tiempo de inferencia medido por página procesada.

El éxito del sistema se considerará alcanzado si se logra una reducción del ≥ 15 % en CER respecto al baseline (EasyOCR) y una latencia media < 3 s/página.

Resultados preliminares

El pipeline implementado y utilizado para la obtención de las métricas iniciales de rendimiento se encuentra documentado en el cuaderno ocr_benchmark_notebook.ipynb, disponible en el repositorio del proyecto.

Este notebook contiene la implementación completa del pipeline descrito, incluyendo la conversión de PDF a imagen, el preprocesado, la ejecución de los motores OCR neuronales (EasyOCR, PaddleOCR y DocTR) y el cálculo de las métricas CER y WER.

Los resultados obtenidos se muestran en la siguiente tabla:

Tabla 1. *Métricas preliminares WER/CER.*

Modelo	***WER***	***CER***	***Comentario***
EasyOCR	0.113	0.057	Mejor equilibrio general entre precisión y velocidad.
PaddleOCR	0.092	0.079	Detección robusta, ligera pérdida de precisión de caracteres.
DocTR	0.195	0.065	Buen reconocimiento a nivel de carácter, menor segmentación de palabras.

Estos resultados confirman la viabilidad técnica del pipeline y permiten establecer una línea base sólida para las siguientes fases, que incluirán optimización de modelos, análisis de latencia y evaluación lingüística del texto extraído.

Plan de trabajo

Tabla 2. *Plan de trabajo.*

Fase	***Periodo estimado***	***Actividad principal***	***Resultado esperado***
1	Semanas 1–3	Diseño e implementación del pipeline OCR multimotor	Sistema funcional inicial
2	Semanas 4–6	Ejecución de pruebas con EasyOCR, PaddleOCR y DocTR	Resultados comparativos iniciales
3	Semanas 7–10	Optimización de modelos y análisis de métricas	Informe de mejora de rendimiento
4	Semanas 11–14	Redacción final, conclusiones y presentación	Documento y demostrador técnico

4. Desarrollo específico de la contribución

En este apartado debes desarrollar la descripción de tu contribución. Es muy dependiente del tipo de trabajo concreto, y puedes contar con la ayuda de tu director para estudiar cómo comunicar los detalles de tu contribución. A continuación, te presentamos la estructura habitual para cada uno de los tipos de trabajo, aunque suele ser común desarrollar los apartados en función de las fases o actividades que se hayan establecido en la metodología de trabajo.

Tipo 1. Piloto experimental

Este tipo de trabajos suelen seguir la estructura típica al describir experimentos científicos, dividida en descripción del experimento, presentación de los resultados y discusión de los resultados.

Capítulo 4 - Descripción detallada del experimento

En el capítulo de Objetivos y Metodología del Trabajo ya habrás descrito a grandes rasgos la metodología experimental que vas a seguir. Pero si tu trabajo se centra en describir un piloto, deberás dedicar un capítulo a describir con todo detalle las características del piloto. Como mínimo querrás mencionar:

- Qué tecnologías se utilizaron (incluyendo justificación de por qué se emplearon y descripciones detalladas de las mismas).
- Cómo se organizó el piloto
- Qué personas participaron (con datos demográficos)
- Qué técnicas de evaluación automática se emplearon.
- Cómo transcurrió el experimento.
- Qué instrumentos de seguimiento y evaluación se utilizaron.
- Qué tipo de análisis estadísticos se ha empleado (si procede).

Capítulo 5 - Descripción de los resultados

En el siguiente capítulo deberás detallar los resultados obtenidos, con tablas de resumen, gráficas de resultados, identificación de datos relevantes, etc. Es una exposición objetiva, sin valorar los resultados ni justificarlos.

Capítulo 6 - Discusión

Tras la presentación objetiva de los resultados, querrás aportar una discusión de los mismos. En este capítulo puedes discutir la relevancia de los resultados, presentar posibles explicaciones para los datos anómalos y resaltar aquellos datos que sean particularmente relevantes para el análisis del experimento.

Tipo 2. Desarrollo de software

En un trabajo de desarrollo de software es importante justificar los criterios de diseño seguidos para desarrollar el programa, seguido de la descripción detallada del producto resultante y finalmente una evaluación de la calidad y aplicabilidad del producto. Esto suele verse reflejado en la siguiente estructura de capítulos:

Capítulo 4 - Identificación de requisitos

En este capítulo se debe indicar el trabajo previo realizado para guiar el desarrollo del software. Esto debería incluir la identificación adecuada del problema a tratar, así como del contexto habitual de uso o funcionamiento de la aplicación. Idealmente, la identificación de requisitos se debería hacer contando con expertos en la materia a tratar.

Capítulo 5 - Descripción de la herramienta software desarrollada

En el caso de **desarrollos de software**, deberían aportarse detalles del proceso de desarrollo, incluyendo las fases e hitos del proceso. También deben presentarse diagramas explicativos

de la arquitectura o funcionamiento, así como capturas de pantalla que permitan al lector entender el funcionamiento del programa.

Capítulo 6 - Evaluación

La evaluación debería cubrir por lo menos una mínima evaluación de la usabilidad de la herramienta, así como de su aplicabilidad para resolver el problema propuesto. Estas evaluaciones suelen realizarse con usuarios expertos.

Tipo 3. Comparativa de soluciones

Este tipo de trabajos suelen seguir la estructura típica de un estudio comparativo, parten de plantear la comparativa a realizar, describen el desarrollo de la misma y analizan los resultados.

Capítulo 4 - Planteamiento de la comparativa

En este capítulo se debe indicar el trabajo previo realizado para identificar el problema concreto a tratar, así como las posibles soluciones alternativas que se van a evaluar. También se deben identificar los criterios de éxito para la comparativa, las medidas que se van a tomar, etc.

Capítulo 5 - Desarrollo de la comparativa

En este capítulo se debería desarrollar con todo detalle la comparativa realizada, con todos los resultados y mediciones obtenidos. Puede ser útil acompañar las descripciones con gráficas, tablas y otros instrumentos para plasmar los datos obtenidos.

Capítulo 6 - Discusión y análisis de resultados

Mientras que el capítulo anterior se centraría en informar de los resultados y comparaciones obtenidos, en este capítulo se abordará la discusión sobre su posible significado, así como el análisis de las ventajas y desventajas de las distintas soluciones evaluadas.

En el capítulo de Objetivos y Metodología del Trabajo ya habrás descrito a grandes rasgos la metodología experimental que vas a seguir. Pero si tu trabajo se centra en describir un piloto, deberás dedicar un capítulo a describir con todo detalle las características del piloto. Como mínimo querrás mencionar:

- Qué tecnologías se utilizaron (incluyendo justificación de por qué se emplearon y descripciones detalladas de las mismas).
- Cómo se organizó el piloto
- Qué personas participaron (con datos demográficos)
- Qué técnicas de evaluación automática se emplearon.
- Cómo transcurrió el experimento.
- Qué instrumentos de seguimiento y evaluación se utilizaron.
- Qué tipo de análisis estadísticos se ha empleado (si procede).

.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Este último capítulo (en ocasiones, dos capítulos complementarios) es habitual en todos los tipos de trabajos y presenta el resumen final de tu trabajo y debe servir para informar del alcance y relevancia de tu aportación.

Suele estructurarse empezando con un resumen del problema tratado, de cómo se ha abordado y de por qué la solución sería válida.

Es recomendable que incluya también un resumen de las contribuciones del trabajo, en el que relaciones las contribuciones y los resultados obtenidos con los objetivos que habías planteado para el trabajo, discutiendo hasta qué punto has conseguido resolver los objetivos planteados.

5.2. Líneas de trabajo futuro

Finalmente, se suele dedicar una última sección a hablar de líneas de trabajo futuro que podrían aportar valor añadido al TFE realizado. La sección debería señalar las perspectivas de futuro que abre el trabajo desarrollado para el campo de estudio definido. En el fondo, debes justificar de qué modo puede emplearse la aportación que has desarrollado y en qué campos.

Referencias bibliográficas

Según la normativa APA debe ponerse con sangría francesa y debe estar ordenado por orden alfabético según el apellido del primer autor.

Toda la bibliografía que aparezca en este apartado debe estar citada en el trabajo. La mayor parte de las citas deben aparecer en el capítulo 2, que es donde se realiza el estudio del estado del arte. Además, se recomienda evitar citas que hagan referencia a Wikipedia y que no todas las referencias sean solo enlaces de internet, es decir, que se vea alguna variabilidad entre libros, congresos, artículos y enlaces puntuales de internet.

Se recomienda encarecidamente utilizar el gestor de bibliografía de Word para gestionar la bibliografía.

Ejemplo:

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. *Management Review (AMA FORUM), 70*, 35-36.

Anexo A. Código fuente y datos analizados

Es recomendable que el estudiante incluya en su memoria la URL del repositorio donde tiene alojado el código fuente desarrollado durante el TFE. El estudiante debe ser el único autor del código y único propietario del repositorio. En el repositorio no debe haber commit de ningún otro usuario del repositorio.

De igual forma, los datos que hayan utilizado para el análisis, siempre que así se considere oportuno, también deberían están alojamos en el mismo repositorio.

Si el TFE está asociado a una actividad o proyecto de Empresa, se debe justificar en la memoria que, por temas de confidencialidad, no se deja disponible ni el código fuente ni los datos utilizados.