EXTRACCIÓN DE INFORMACIÓN EXPERIMENTAL DE RENDIMIENTO Y SÍNTESIS DE CELDAS SOLARES DE PEROVSKITA MEDIANTE MODELOS DE PROCESAMIENTO DEL LENGUAJE NATURAL

Mary Zuleika Jiménez Díaz

Trabajo de Grado para optar al título de Magíster en Ingeniera Electrónica

Director

Franklin Alexander Sepúlveda Sepúlveda

PhD en Ingeniería - Automática

Codirector

Mónica Andrea Botero Londoño

PhD en Ciencias-Físicas

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2026

Dedicatoria

Este trabajo esta dedicado a mi familia y a quienes estuvieron presentes incluso en los momentos más difíciles de este camino académico.

Agradecimientos

A mi familia, por el apoyo emocional y la paciencia durante este proceso.

A mis asesores y los grupos investigación CEMOS y GISEL por el acompañamiento y facilitar el acceso a la Workstation utilizada durante el desarrollo experimental.

Este trabajo también representa el resultado de múltiples horas de aprendizaje autónomo, experimentación y perseverancia.

Tabla de Contenido

Introducción **24**

1. Objetivos **32**

1.1. Objetivo general 32

1.2. Objetivos específicos 32

2. Fundamentos de las Celdas solares de perovskita **33**

2.1. Caracterización de celdas solares de Perovskita 34

2.2. Fabricación de celdas solares de Perovskita 35

2.3. La ciencia computacional en el descubrimiento de materiales 37

3. Procesamiento de Lenguaje Natural para La extracción de información **39**

3.1. Procesamiento de Lenguaje Natural (Natural Language Processing, NLP) 39

3.2. Extracción de información (Information Extraction, IE) 40

3.3. Arquitectura de la IE 41

3.4. Modelos de Ingeniería del Conocimiento Knowledge Engineering, KE) 43

3.5. Modelos de Aprendizaje Profundo (Deep Learning, DL) 43

3.6. Tipos de aprendizaje de entrenamiento y adaptación de Modelos 44

3.7. Estrategias de entrenamiento y adaptación de Modelos 45

3.8. Técnicas del Aprendizaje por transferencia (Transfer Learning) 46

3.8.1. Ajuste Fino (Fine-Tuning) 49

3.9. Modelos de Lenguaje Preentrenados (Pre-trained Language Models, PLMs) 50

3.10. Reconocimiento de entidades nombradas (Named Entity Recognition, NER) 51

3.11. Extracción de relaciones (Relation Extraction, RE) 52

3.12. Desarrollo de modelos NLP 55

3.12.1. Representación de los datos 57

3.12.2. Entrenamiento 58

3.12.3. Hiperparámetros de Entrenamiento 59

3.12.4. Regularización 60

3.12.5. Técnicas complementarias 60

3.12.6. Evaluación 61

3.12.7. Validación 63

3.12.7.1. Hold-Out 63

3.12.7.2. Validación cruzada tipo K-Fold (k-Fold Cross Validation) 64

3.13. Recolección de datos de la Web (Web Data collection) 65

4. Estado del Arte 68

4.1. NLP convencional 68

4.2. NLP con Deep learning 68

4.3. Modelos PLMS basados en arquitectura Tranformers 69

4.4. NLP para la extracción de información y análisis de artículos científicos 69

4.5. Transfer learning en modelos PLMs para la extracción de información 72

4.6. NLP para la extracción y análisis de información aplicado a celdas solares de perovskita 73

5. Diseño del Pipeline de extracción y estructuración de información aplicado a celdas solares de Perovskita 76

5.1. Selección y Recolección de los datos 79

5.1.1. Selección del Dataset 79

5.1.2. Recolección de los datos mediante Web Scraping 80

5.2. Preprocesamiento de los datos 82

5.3. Selección de modelos 83

5.3.1. Selección del modelo NER 84

5.3.2. Selección del modelo RE 85

5.4. Generación del conjunto de datos etiquetados 86

5.4.1. Generación de la primer versión del Dataset para la tarea NER 87

5.4.2. Depuración del Dataset para la tarea NER 88

5.4.3. Conversión y depuración del Dataset para la tarea RE 90

5.4.4. Generación del Dataset para la evaluación (Test) independiente 91

5.5. Entrenamiento de modelos Deep Learning para Information Extraction 91

5.5.1. Evaluación mediante un conjunto de prueba (Test) independiente 93

5.6. Entrenamiento del modelo para el Reconocimiento de entidades nombradas (NER) 93

5.7. Entrenamiento del modelo para la Extracción de relaciones (RE) 95

5.8. Inferencia y extracción automática de información 98

5.8.1. Extracción del texto y Reconocimiento de entidades nombradas (NER) 98

5.8.2. Extracción de relaciones (RE) 100

5.9. Post-Procesamiento basado en Ingeniería del Conocimiento 101

5.10. Validación de los Resultados 104

5.10.1. Comparación a nivel de entidad 104

5.10.2. Comparación a nivel de device 105

5.11. Entorno Experimental 105

6. Resultados **107**

6.1. Web Scraping para descargar artículos 107

6.2. Construcción del conjunto de datos etiquetados (Dataset) para los modelos NER y RE 109

6.2.1. Construcción del Dataset para la evaluación (Test) independiente de los modelos NER y RE 112

6.3. El modelo NER 113

6.3.1. Evaluación del modelo sobre el conjunto de validación 115

6.3.2. Validación cruzada del modelo 117

6.3.3. Evaluación (Test) independiente del modelo 118

6.4. Selección de modelo RE 121

6.4.1. Comparación de Arquitecturas 121

6.4.1.1. Basado en Características (Feature based) 121

6.4.1.2. Ajuste Fino (Fine Tuning) 122

6.4.2. Comparación de Modelos 122

6.4.3. Análisis 124

6.5. El modelo RE 125

6.5.1. Matriz de hiperparámetros 125

6.5.2. Variación de la semilla aleatoria 126

6.5.3. Uso del Balanceo de clases 127

6.5.4. Resultado de los experimentos 128

6.5.5. Evaluación del modelo sobre el conjunto de validación 129

6.5.6. Validación cruzada del modelo 131

6.5.7. Evaluación (Test) independiente del modelo 132

6.6. Construcción automática de la Base de datos (Database) a partir de la fase de inferencia 134

6.7. Post-procesamiento: la Reconstrucción de devices 136

6.8. Validación de los Resultados 137

6.8.1. Comparación a nivel de entidad 138

6.8.2. Comparación a nivel de devices 139

6.8.3. Análisis de un artículo 140

6.9. Limitaciones 143

6.10. Disponibilidad de datos 144

7. Conclusiones **145**

Referencias Bibliográficas **148**

Apéndices **169**

Lista de Figuras

Figura 1. Capacidad de potencia instalada y adicional anual de energía renovable. 25

Figura 2. Evolución del número de documentos sobre celdas fotovoltaicas de Perovskitas publicadas. 26

Figura 3. Representación gráfica de la estructura y las capas contenidas en una celda solar de Perovskita. 34

Figura 4. Esquema del enfoque tecnológico para el descubrimiento de materiales y/o estructuras. 38

Figura 5. Diagrama de la distribución general del procesamiento natural del lenguaje (NLP) como parte de la inteligencia artificial. 39

Figura 6. Diagrama general de la arquitectura de la extracción de información, IE. 42

Figura 7. Ilustración de los métodos de *Transfer Learning* más comúnmente aplicados en el NLP. 47

Figura 8. Ejemplo de la aplicación de la tarea NER. 52

Figura 9. Ejemplo de la aplicación de la tarea RE. 53

Figura 10. Esquema general del proceso interno de aprendizaje de un modelo de aprendizaje profundo (*Deep Learning*, DL). 59

Figura 11. Diagrama resumen del *Pipeline* (Tubería) desarrollado. 76

Figura 12. Representación conceptual del proceso selección y Recolección de los datos. 79

Figura 13. Representación conceptual del preprocesamiento de los datos. 82

Figura 14. Representación conceptual del proceso de selección de los modelos NER y RE. 84

Figura 15. Representación conceptual del proceso de generación del conjunto de datos etiquetados para los modelos NER y RE. 87

Figura 16. Ejemplo de una oración con sus entidades nombradas del *dataset* de entrenamiento NER. 88

Figura 17. Representación grafica del proceso de corrección y mejora de los datos etiquetados del *Dataset*. 89

Figura 18. Ejemplo de una oración con su pareja entidades-relacion del *Dataset* de entrenamiento RE. 90

Figura 19. Representación conceptual del proceso de entrenamiento de los modelos NER y RE. 92

Figura 20. Esquema general del Proceso de selección, optimización y validación del modelo. 92

Figura 21. Representación conceptual del proceso de inferencia y extracción automática de información. 98

Figura 22. Ejemplo de la estructura de representación utilizada en los resultados de la inferencia del modelo NER. 99

Figura 23. Ejemplo de la estructura de representación utilizada en los resultados de la inferencia del modelo RE. 100

Figura 24. Representación conceptual del post-procesamiento basado en Ingeniería del Conocimiento. 101

Figura 25. Representación de las métricas complementarias. 102

Figura 26. Ejemplo de la estructura de representación utilizada en los resultados del post-procesamiento. 103

Figura 27. Representación conceptual del proceso de validación de los resultados. 104

Figura 28. Gráfica de las curvas de entrenamiento del modelo NER (*MODEL_ner*) seleccionado. 115

Figura 29. Gráfica de las curvas de entrenamiento del modelo RE (*MODEL_re*) seleccionado. 129

Figura 30. Ventana inicial de la API. 175

Figura 31. Ventana principal de la API. 176

Figura 32. Ventana secundaria de la API. 177

Figura 33. Tabla exitosamente extraída del artículo $HTMLs_620.html$. 181

Figura 34. Tabla erróneamente extraída del artículo $PDFs_1207.pdf$. 181

Figura 35. Curvas de entrenamiento de las 16 combinaciones evaluadas para el modelo NER. 190

Figura 36. Curvas de entrenamiento y pérdida (*loss*) obtenidas durante el proceso de validación cruzada por k-Fold, con $K = 5$. 191

Lista de Tablas

Tabla 1. Principales tipos de aprendizaje utilizados en ML/DL. 45

Tabla 2. Estrategias de entrenamiento y adaptación de modelos de ML/DL. 46

Tabla 3. Resumen de las ventajas y desventajas de los dos enfoques para aplicar la tarea RE. 55

Tabla 4. Principales características y detalles de las principales herramientas, bibliotecas y modelos de NLP. 56

Tabla 5. Comparación entre métricas macro y micro. 63

Tabla 6. Comparación entre KFold y StratifiedKFold. 64

Tabla 7. Entidades nombradas definidas para la extracción de información en celdas solares de Perovskita. 77

Tabla 8. Relaciones definidas para la extracción de información en celdas solares de Perovskita. 78

Tabla 9. Principales versiones de modelo preentrenado SciBERT. 86

Tabla 10. Configuración del mejor modelo NER, llamado "*MODEL_ner*". 95

Tabla 11. Configuración del mejor modelo RE, llamado "*MODEL_re*". 97

Tabla 12. Entornos en los que se experimento y trabajo el *Pipeline*. 106

Tabla 13. Resumen del proceso de Selección y Recolección de los datos. 108

Tabla 14. Distribución del *Dataset* para la tarea NER. 110

Tabla 15. Distribución del *Dataset* para la tarea NER balanceados. 110

Tabla 16. Distribución del *Dataset* de los datos etiquetados para la tarea RE. 111

Tabla 17. Distribución del *Dataset* usado en la evaluación independiente del modelo NER. 112

Tabla 18. Distribución del *Dataset* usado en la evaluación independiente del modelo RE. 113

Tabla 19. Resumen de las 16 combinaciones evaluadas para el modelo NER. 114

Tabla 20. Métricas del modelo *MODEL_ner* en la evaluación sobre el conjunto de validación. 116

Tabla 21. Métricas promedio por entidad del modelo *MODEL_ner* sobre el conjunto de validación. 117

Tabla 22. Métricas promedio de la validación cruzada por k-Fold del modelo *MODEL_ner*, con $K = 5$. 118

Tabla 23. Desempeño del modelo *MODEL_ner* sobre un *Test* independiente. 119

Tabla 24. Desempeño promedio por entidad del modelo *MODEL_ner* sobre un *Test* independiente. 120

Tabla 25. Resumen de las Métricas de evaluación macro-average para la lista de modelos preentrenados. 124

Tabla 26. Resumen de las Métricas de evaluación Macro-average_all para diferentes semillas aleatorias. 126

Tabla 27. Distribución balanceada por sobremuestreo del *Dataset* para la tarea RE. 127

Tabla 28. Comparación de las Métricas de evaluación del modelo RE, Macro-average_all, para el entrenamiento con y sin balanceo. 127

Tabla 29. Métricas de evaluación del modelo *MODEL_re* sobre el conjunto de validación. 130

Tabla 30. Métricas promedio obtenidas en validación cruzada por k-Folds estratificados del modelo *MODEL_re*, con $K = 10$. 131

Tabla 31. Desempeño del modelo *MODEL_re* sobre un *Test* independiente. 133

Tabla 32. Resumen de la comparación y validación a nivel de entidades. 138

Tabla 33. Valores promedio de la comparación y validación a nivel de entidades. 138

Tabla 34. Valores promedio de la comparación y validación a nivel de devices. 139

Tabla 35. Comparación de las entidades extraídas del artículo *HTMLs*_620.*html* vs el Database manual. 141

Tabla 36. Comparación de un device reconstruido del artículo *HTMLs*_620.*html* vs los valores encontrados en el Database manual. 141

Tabla 37. Métricas de coherencia calculadas para el dispositivo de la Tabla 36. 142

Tabla 38. Tabla extraída del artículo *HTMLs*_620.*html*. 142

Tabla 39. Resumen del desempeño de los modelos NER y RE sobre los conjuntos de validación y *Test* independiente. 146

Tabla 40. Resumen de las métricas macro de evaluación de las 108 combinaciones evaluadas para el modelo RE. 192

Tabla 41. Correspondencia entre las secciones metodológicas y las carpetas implementadas. 205

Tabla 42. Correspondencia entre las secciones metodológicas y los principales archivos utilizados. 205

Tabla 43. Correspondencia entre las secciones metodológicas y los módulos implementados. 206

Lista de Apéndices

pág.

Apéndice A. Descripción detallada del proceso de generación del conjunto de datos etiquetados 169

Apéndice B. API para la corrección de conjuntos de datos en formato JSON para el entrenamiento, evaluación y validación de modelos NER de spaCy 175

Apéndice C. Proceso de selección, optimización y validación del modelo aplicado para el entrenamiento de un modelo *Deep Learning* usando aprendizaje supervisado en Python. 178

Apéndice D. Extracción automática de la información de tablas artículos científicos en formato PDF y HTML 180

Apéndice E. Descripciones de las estructuras de datos usadas 182

Apéndice F. Descripción detallada del proceso de post-procesamiento basado en Ingeniería del Conocimiento 187

Apéndice G. Gráficas 190

Apéndice H. Tabla Resumen de los resultados de la Matriz de Hiperparámetros del proceso del modelo RE. 192

Apéndice I. Organización de los archivos y Códigos 205

Apéndice J. Artículo derivado de la investigación 208

Glosario

Aprendizaje profundo (*Deep learning*) se puede describir como la nueva evolución del aprendizaje automático. Se trata de un algoritmo automático que imita la percepción humana inspirada en nuestro cerebro y la conexión entre neuronas. El DL es la técnica que más se acerca a la forma en la que aprendemos los humanos (Bismart, 2021).

Bias (sesgos, *b*) es un término adicional que permite desplazar la función de activación.

Clasificación del texto (*Text classification*) es una técnica de aprendizaje automático que asigna un conjunto de categorías predefinidas al texto abierto. Se puede utilizar para organizar, estructurar y categorizar prácticamente cualquier tipo de texto, desde documentos, estudios médicos y archivos, y en toda la web (MonkeyLearn, 2021).

ETL (*Electron Transport Layer*) es la capa de transporte de electrones en una celda solar de perovskita (PSC). Su función es extraer y transportar los electrones fotogenerados desde la capa absorbente hacia el electrodo correspondiente, al tiempo que bloquea el transporte de huecos y reduce las pérdidas por recombinación.

Expresiones regulares (*Regular expression*) es una secuencia de caracteres finita que define un patrón de búsqueda. Una secuencia de expresión regular se compone de caracteres simples, como /abc/, o una combinación de caracteres simples y especiales (Nagy, 2018). El lenguaje Python usa el módulo “re” (Python Software Foundation, 2021).

FF (*Fill Factor*) es un parámetro que describe la forma de la curva corriente-voltaje de una celda solar. Se define como la relación entre la potencia eléctrica máxima de salida y el producto del voltaje de circuito abierto (V_{oc}) y la corriente de cortocircuito (J_{sc}) (C.B.Honsberg and S.G.Bowden, 2019).

HTL (*Hole Transporting Layer*) es la capa de transporte de huecos en una celda solar de perovskita (PSC). Su función es extraer y transportar los huecos fotogenerados desde la capa absorbente hacia el electrodo correspondiente, al tiempo que bloquea el transporte de electrones y reduce las pérdidas por recombinación.

JSC (*Short-Circuit Current*) es la corriente de la celda solar cuando el voltaje a través de la celda solar es cero (es decir, cuando la celda solar está en cortocircuito) (C.B.Honsberg and S.G.Bowden, 2019).

Lematización (*Lemmatization*) es el proceso de reducir una palabra a su raíz o unidad base, normalmente previo al desarrollo del procesamiento de datos. Implica el uso de su forma de diccionario o forma de lema (palabra en forma canónica) de una palabra. Para que la lematización resuelva una palabra en su lema, necesita conocer su parte del discurso (Heidenreich, 2018).

Minería de datos (*Data Mining*) es un conjunto de técnicas y procedimientos que se pueden desarrollar a partir de fuentes de datos transformados en archivos planos sin formatos, que están hechos de este análisis predictivo, utilizando técnicas de estudio estadístico para predecir o anticipar estadísticas y medidas de certeza basadas en hechos existentes (Manjarres et al., 2018).

NER (*Named Entity Recognition*) es una subtarea de extracción de información, que intenta reconocer y categorizar entidades nombradas en texto no estructurado en categorías predefinidas como los nombres de personas, organizaciones y ubicaciones (Moradi et al., 2017).

PCE (*Power Conversion Efficiency*) es la relación entre la potencia eléctrica máxima generada por una celda solar y la potencia de la radiación solar incidente sobre su superficie. Este parámetro cuantifica la capacidad del dispositivo para convertir la energía solar en energía eléctrica y depende tanto de las características de la celda como de las condiciones de operación, tales como el espectro e intensidad de la luz incidente y la temperatura (C.B.Honsberg and S.G.Bowden, 2019).

Pesos (*Weights*, *w*) son los valores que multiplican las entradas en cada conexión entre neuronas.

Pipeline (*Tubería*) es un término inglés que puede traducirse como "tubería". En el entorno de la informática se refiere al proceso de segmentar datos para incrementar el rendimiento de un sistema; es un flujo de datos en donde la salida de una fase o segmento es la entrada de otra. Se emplea normalmente en microprocesadores, tarjetas gráficas y software (Porto and Merino, 2015).

Procesamiento natural del lenguaje (*NLP, Natural Language Processing*) actualmente se define como una sub-área de la inteligencia artificial que se centra en la interpretación automática del lenguaje humano, es decir, procesar la información comunicada, no simplemente las letras o sonidos del lenguaje. Es una herramienta invaluable en el análisis, agregación y simplificación de texto (Kang et al., 2020).

PTF (*Perovskite Layer*) es la capa absorbente de una celda solar de perovskita (PSC). Su función es absorber la radiación incidente y generar pares electrón-hueco, los cuales son posteriormente separados y transportados hacia las capas ETL y HTL para su recolección en los electrodos.

RE (*Relation Extraction*) es una subtarea de extracción de información que intenta identificar y clasificar las relaciones entre las entidades detectadas por la subtarea NER. Busca encontrar relaciones a partir de un texto plano, organizando información no estructurada en información estructurada (Wang et al., 2022).

VOC (*Open-Circuit Voltage*) corresponde al voltaje medido entre los terminales de una celda solar bajo condición de circuito abierto, es decir, cuando la corriente de salida es nula ($I = 0$). Este parámetro está determinado por el balance entre la corriente fotogenerada y los mecanismos de recombinación presentes en el dispositivo (C.B.Honsberg and S.G.Bowden, 2019).

Resumen

Título: EXTRACCIÓN DE INFORMACIÓN EXPERIMENTAL DE RENDIMIENTO Y SÍNTESIS DE CELDAS SOLARES DE PEROVSKITA MEDIANTE MODELOS DE PROCESAMIENTO DEL LENGUAJE NATURAL [*]

Autor: Mary Zuleika Jiménez Díaz [**]

Palabras Clave: Celdas solares de Perovskita, Procesamiento de lenguaje natural (NLP), Modelos de Lenguaje Preentrenado (PLMs), Modelos de Lenguaje a Gran Escala (LLM), Recopilación de datos, Minería web, Aprendizaje Automático, Extracción de información.

Descripción: La crisis energética iniciada en 2021 ha impulsado la búsqueda de fuentes alternativas de energía limpia, destacándose la energía solar por su crecimiento sostenido. En este contexto, las celdas solares de Perovskita representan una tecnología prometedora debido a su alta eficiencia y facilidad de fabricación. Sin embargo, el rápido aumento de publicaciones científicas ha generado grandes volúmenes de información compleja y diversa, dificultando su análisis y aprovechamiento.

Los métodos tradicionales de identificación de tendencias y relaciones en la literatura científica usados para desarrollar celdas solares son lentos y demandan un esfuerzo considerable. Para afrontar este desafío, se emplean técnicas de procesamiento de lenguaje natural (NLP), recopilación de datos y aprendizaje por transferencia (Transfer Learning), que permiten adaptar modelos preentrenados al lenguaje científico especializado.

Este proyecto desarrolla un Pipeline híbrido basado en dos modelos NLP usando Transfer Learning, NER y RE, complementados con reglas heurísticas para la extracción contextualizada de información. El sistema agrupa entidades en dispositivos (devices) que integran las capas ETL, HTL y Perovskita, parámetros de desempeño (Voc, Jsc, PCE y FF) y métodos de síntesis. Con énfasis en minimizar los requisitos de recursos computacionales necesarios para desarrollar e implementar el Pipeline.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Franklin Alexander. PhD en Ingeniería-Automática. Codirector: Mónica Andrea Botero Londoño. PhD en Ciencias-Físicas.

Abstract

Title: EXTRACTION OF EXPERIMENTAL PERFORMANCE AND SYNTHESIS INFORMATION FOR PEROVSKITE SOLAR CELLS USING NATURAL LANGUAGE PROCESSING MODELS [*]

Author: Mary Zuleika Jiménez Díaz [**]

Keywords: Perovskite Solar Cells, Natural Language Processing (NLP), Pre-trained Language Models (PLMs), Large Language Models (LLM), Data Collection, Web Scraping, Machine Learning, Information Extraction.

Description: The energy crisis since 2021 has driven the search for alternative sources of clean energy, with solar energy standing out for its sustained growth. In this context, Perovskite solar cells represent a promising technology due to their high efficiency and ease of manufacture. However, the rapid increase in scientific publications has generated large volumes of complex and diverse information, hindering its analysis and utilization.

Traditional methods for identifying trends and relationships in the scientific literature used to develop solar cells are slow and require considerable effort. To overcome these challenges, Natural Language Processing (NLP), data collection, and Transfer Learning techniques are employed, allowing pre-trained models to be adapted to specialized scientific language.

This project develops a hybrid pipeline based on two NLP models using transfer learning, NER and RE, complemented by heuristic rules for contextualized information extraction. The system groups entities into devices that integrate the ETL, HTL, and Perovskite layers, performance parameters (Voc, Jsc, PCE, and FF), and synthesis methods. With an emphasis on minimizing computational resource requirements to develop and implement the Pipeline.

---

[*] Degree Work

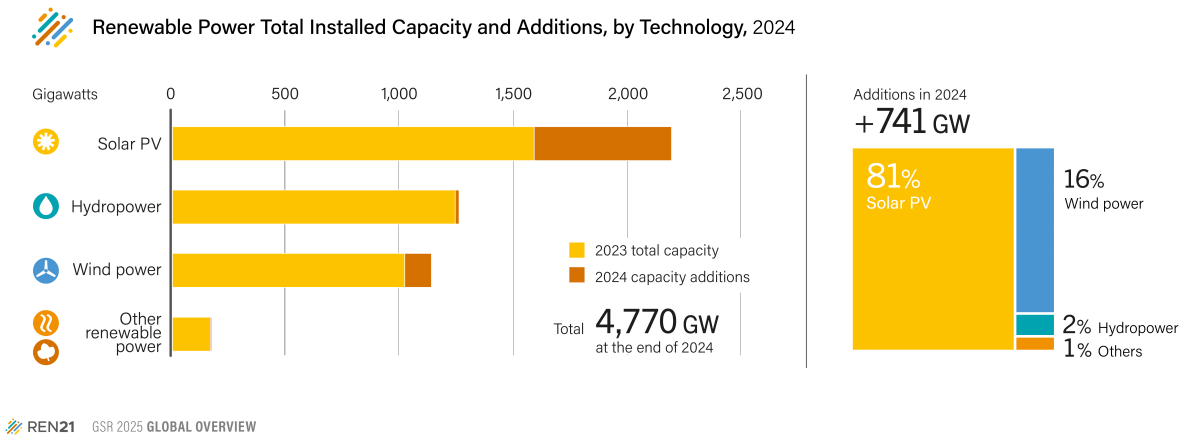
[**] Faculty of Physical-Mechanical Engineering. School of Electrical, Electronic and Telecommunications Engineering. Director: Franklin Alexander. PhD in Engineering-Automation. Co-director: Mónica Andrea Botero Londoño. PhD in Physical Sciences.

Introducción

En el contexto actual, el desarrollo energético ha promovido las energías renovables de bajo impacto ambiental, las cuales han tenido un crecimiento significativo en años recientes. Dentro de estas, la energía fotovoltaica ha experimentado un aumento significativo: la capacidad superó la marca de los 2 terawatts[TW] en 2024, alcanzando los 2,25[TW], casi el doble del total mundial en 2022. REN21(2025) reporta que en 2024 se agregaron 602 [GW] de nueva capacidad, un 32 % más que en 2023 (S. REN21, 2025) (ver Fig. 1). Aunque existen variadas tecnologías de celdas solares, las celdas solares de Perovskita (con valores de eficiencia energética de hasta el 27.3 % según (National Laboratory of the Rockies, 2026) y 25.7 % según (Liu et al., 2024; Yang et al., 2024; Zhang et al., 2024)) se destacan debido a su eficiencia en constante mejora, excelentes propiedades ópticas y eléctricas, montaje sencillo, peso ligero, rentabilidad y a su facilidad de manejo, teniendo el potencial de superar a otras tecnologías fotovoltaicas en el futuro (Elangovan et al., 2024). Sin embargo, los métodos tradicionales para el descubrimiento y desarrollo de nuevos materiales suelen requerir procesos iterativos de síntesis, caracterización y validación experimental que demandan una cantidad considerable de tiempo y recursos (Himanen et al., 2019).

Figura 1

Capacidad de potencia instalada y adicional anual de energía renovable.



Nota. Tomado de *Renewables 2025 Global Status Report* de S. REN21 (2025).

El interés por la investigación de celdas solares de Perovskita (PSCs) se ha incrementado notablemente. Este interés se ha visto reflejado en el enorme incremento de artículos científicos que incluyen la palabra clave perovskita, según los datos recopilados de bases de datos como Scopus y Web of Science, especialmente desde que se propuso la celda solar de perovskita en el 2013 (Tao et al., 2021). De hecho, entre los años 2008 a 2026 aproximadamente se han publicado 96237 documentos sobre Perovskitas fotovoltaicas (ver Fig. 2). A medida que estas investigaciones se multiplican, la complejidad y diversidad de la información también aumenta, dificultando la extracción eficiente y precisa de datos relevantes de los artículos científicos.

Figura 2

Evolución del número de documentos sobre celdas fotovoltaicas de Perovskitas publicadas.



	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
Scopus	87	437	1183	2036	2565	3308	3740	4134	4364	5054	5394	6160	6750	3009
Web of science	54	447	1212	2072	2803	3487	3957	4376	4957	5054	5319	6045	6374	1741

Nota. Datos recopilados en las bases de datos Scopus y Web of Science, hasta abril de 2026.

Una estrategia para acelerar el desarrollo de nuevas composiciones de perovskita y el diseño de dispositivos fotovoltaicos consiste en incorporar técnicas de aprendizaje automático (*Machine Learning*, ML). En los últimos años, estas técnicas han despertado un notable interés en la comunidad científica, como lo evidencia el creciente número de publicaciones dedicadas a su aplicación en celdas solares de Perovskita (Chen et al., 2024; Kadulkar et al., 2022; Rabiee, 2023; Thilakaratne et al., 2019; Zhu et al., 2023; Yin et al., 2024).

El desempeño de los modelos de aprendizaje automático depende en gran medida de la disponibilidad y calidad de los datos utilizados para su entrenamiento. Estos datos pueden provenir de

experimentos, simulaciones computacionales, bases de datos especializadas o información reportada en la literatura científica. En particular, la generación de nuevos datos experimentales suele requerir procesos de síntesis, fabricación y caracterización que pueden demandar una cantidad considerable de tiempo y recursos. En este contexto, aprovechar el conocimiento acumulado en la creciente cantidad de artículos científicos publicados sobre celdas solares de Perovskita constituye una alternativa atractiva para apoyar el descubrimiento y optimización de nuevas composiciones, materiales y arquitecturas de dispositivos, contribuyendo potencialmente a reducir los tiempos y costos asociados a su desarrollo.

Existen algunas iniciativas en las que se han recolectado datos de PSCs manualmente. En particular, en Li et al. (2019a) se toman datos de 333 PSCs a partir de 200 artículos científicos. Luego, este conjunto de datos es complementado en un trabajo posterior (Vélez et al., 2023) al agregarle información acerca del espesor y del tamaño de grano de la capa de perovskita [1]. Sin embargo, en ambos casos la cantidad de datos es limitada. De otra parte, se reporta un conjunto de datos de Perovskita más grande y completo en (Jacobsson et al., 2022), en donde se recopilaron más de 400 variables de más de 42000 registros de celdas solares de Perovskita entre los años 2003 a 2020 [2]. A pesar de contar con este conjunto de datos, es importante actualizar la información y estar realizando este trabajo manualmente es una tarea ardua debido a la notable cantidad de artículos que se publican anualmente.

---

[1] Los datos se pueden descargar desde `https://data.mendeley.com/datasets/bbybhrz4ny/2`

[2] Los datos se pueden descargar desde `https://www.perovskitedatabase.com/`

Sin embargo, la gran cantidad de información y su constante actualización hacen que la recolección manual de la información relevante, sea una tarea ardua e intensiva en tiempo y costos en mano de obra calificada. En este contexto, el Procesamiento de Lenguaje Natural (*Natural Language Processing*, NLP) (Zaman et al., 2020) emerge como una herramienta de notable utilidad, la cual es una tecnología que se ocupa del lenguaje humano y se aplica en diversos campos para gestionar una amplia gama de ideas e investigaciones expresadas lingüísticamente (Lee et al., 2023; Liddy, 2001). Entre sus principales aplicaciones están, Extracción de Información (*Information Extraction*, IE), Respuesta a preguntas (*Question Answering*), Traducción automática (*Machine Translation*), entre otras aplicaciones. En particular, con las técnicas de IE (Abdullah et al., 2023) es posible acceder a la información relacionada a la estructura, métodos de fabricación, condiciones de síntesis y las variables de desempeño de celdas solares de perovskita contenidas en documentos.

Existen variados enfoques para la implementación de sistemas NLP. Anteriormente se utilizaron sistemas con representaciones basadas en N-gramas, *embeddings* (word2vec, ...) y bolsas de palabras a manera de representación; mientras que, a nivel de modelado se utilizaron técnicas como máquinas de vectores de soporte (SVM), redes neuronales (RNN) y Cadenas Ocultas de Markov. Recientemente surgió el aprendizaje profundo (*Deep Learning*, DL) con modelos más complejos como Redes Neuronales Recursivas, Convolucionales y los Modelos de Lenguaje Preentrenados (*Pre-trained Language Models*, PLMs) (Fan et al., 2023). Hoy en día el estado del arte en cuanto a NLP corresponde a los *Transformers* (Wankmüller, 2022).

En trabajos previos se han diseñado y adaptado sistemas de NLP de *Deep Learning* para la extracción automática de información a partir de artículos científicos en áreas del conocimiento

tales como la tecnología (Li et al., 2019b), la medicina (Hu et al., 2024a; Yu et al., 2024), la ciencia de materiales (Beard and Cole, 2022; Hong, 2023) y la química (Krallinger et al., 2017). En lo referente a la utilización de NLP para celdas solares de perovskita, los trabajos previos son escasos.

Algunos de estos trabajos se exponen en artículos como: Jiménez-Díaz et al. (2022) tesis de pregrado, un trabajo previo donde se presenta un algoritmo de reconocimiento de entidades nombradas (*Named Entity Recognition*, NER) para extraer información relacionada con celdas solares de Perovskita, haciendo uso del modelo de redes neuronales convolucionales, SpaCy. El trabajo se limita al entrenamiento y desarrollo de un modelo NER, es decir se entrena un modelo que solo se aplica para la tarea NER, que no consideraba la extracción de relaciones ni da contexto a la información extraída.

Luego, en Beard and Cole (2022) se genera un conjunto de datos estructurados relacionados con PSCs, usando el modelo de redes neuronales convolucionales, *ChemDataExtractor*. También se limita a la extracción de NER y hace uso de la modificación de un modelo ya existente, generando un conjunto de datos con la información que encontraron disponible en la web a través de una búsqueda simple en Elsevier y la Real Sociedad de Química (RSC), teniendo que limitarse en muchos casos a solo extraer información del Abstract de los artículos.

En Xie et al. (2024) generan bases de datos usando los Modelos de Lenguaje a Gran Escala (*Large language models*, LLMs) que hacen parte de los PLMs, y redes convolucionales. Usan modelos basados en *Transformers* (Wolf et al., 2020a), que son modelos LLMs de código abierto (como GPT, BERT, BART, etc) que desarrollan una comprensión estadística del lenguaje en el

que han sido entrenados, que actualmente esta en auge, pero que junto a otros modelos de *Deep Learning*, tienen como desventajas:

- Una alta complejidad de los modelos, que hace tiendan a sobre-modelarse.
- Altos costos de tiempo del etiquetado en el reconocimiento de Entidades Nombradas (NER) / Relación entre Entidades (RE), para crear los conjuntos de entrenamiento.
- Alto consumo de tiempo en el entrenamiento, grandes costos de hardware (costoso computacionalmente) y la dificultad para acceder a datos relevantes.

Con el fin de minimizar las problemáticas anteriores recientemente se plantea la utilización de modelos PLMs, junto al proceso llamado aprendizaje por transferencia (*Transfer Learning*, TL), a fin de obtener modelos NLP mejor adaptados a una problemática en una disciplina en particular (Yuan, 2022), que logren buenos resultados y minimicen sus desventajas, así como su impacto ambiental. Llegando a referirse a que los dos pilares principales de la NLP moderna: son la evolución de los modelos de lenguaje para ser adaptados entre diferentes aplicaciones (*Transfer Learning*) y la sorprendente eficacia empírica de las arquitecturas *Transformers* (*Deep Learning*) (Lauriola et al., 2022).

Para abordar estos desafíos en la extracción información contextualizada de capas, parámetros de desempeño y síntesis de celdas solares de Perovskita fotovoltaicas, se hace el uso del enfoque de *Transfer learning* aplicado a modelos NER y RE en *Pipeline*, y la combinación de diversas herramientas de NLP, recolección de datos (*data collection*), minería de web(*web scraping*) y reglas heurísticas. Se desarrolla una serie códigos que permiten la extracción de información

aun bajo costo computacional, que permite la recolección por conjuntos de datos o dispositivos de celdas, a partir de cientos y/o miles de artículos científicos. El insumo de entrada del algoritmo corresponde a 1307 de artículos en formato HTML y PDF; y, entrega a modo de salida diferentes archivos JSON, EXCEL y TXT con los resultados, con énfasis en minimizar los requisitos de recursos computacionales necesarios para desarrollar e implementar el *Pipeline*.

Los capítulos dos, tres y cuatro se centran en el desarrollo de la contextualización de los temas, el método implementado en el proyecto y el estado del arte, respectivamente. En el capitulo cinco se describe el método planteado, un *Pipeline* Híbrido conformado por ocho partes : la selección y recolección de los datos, el preprocesamiento de datos, la selección de modelos preentrenados, el entrenamiento de los modelos para la tarea NER y RE, la fase de inferencia de los modelos sobre los datos, un post-procesamiento y la validación de los resultados. El sexto capítulo se centra en presentar y analizar los resultados, definir las limitaciones del método del capítulo cinco y definir la disponibilidad de los datos, en el caso de la aplicación del *Pipeline* sobre las capas, los parámetros de desempeño y los métodos de fabricación de artículos científicos de celdas solares de Perovskita (PSC) en formato PDF y HTML. Luego el capitulo séptimo presenta las conclusiones. Además se incluyen diez apéndices en donde se amplia la información del trabajo.

1 Objetivos

1.1 Objetivo general

Desarrollar un modelo de procesamiento del lenguaje natural mediante aprendizaje por transferencia para extraer información relacionada con los parámetros relacionados al desempeño, el proceso de síntesis y la estructura de las celdas solares de perovskita.

1.2 Objetivos específicos

1. Desarrollar un algoritmo de recolección de datos de la web, para información semi-estructurada y no estructurada de artículos científicos en formato HTML y XML, teniendo como insumo la dirección web del articulo completo.

2. Diseñar un método para la obtención de un modelo de procesamiento de lenguaje natural, mediante aprendizaje por transferencia aplicado a las capas relacionadas con la extracción de entidades y la detección de relación entre entidades, para la extracción contextualizada de los parámetros relacionados.

3. Evaluar el desempeño del método, mediante medidas apropiadas, utilizando como referencia conjuntos de datos obtenidos manualmente por investigadores del área de ciencia de materiales fotovoltaicos.

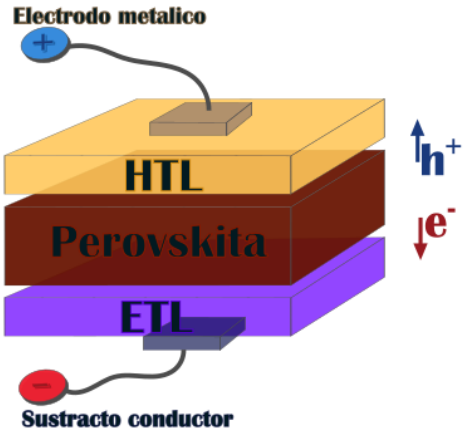
2 Fundamentos de las Celdas solares de perovskita

Las celdas solares de Perovskita son una tecnología de tercera generación, donde el material usado en su capa absorbente es la perovskita (Shah et al., 2023). Se llama perovskita a todo material con la misma estructura cristalina que el titanato de calcio, con la formula ABX_3 (Rojas and Córdova, 2021), su nombre fue dado en honor al mineralogista ruso Lev Perovski y descubierto por el geólogo alemán Gustav Rose en 1839 (Rojas and Córdova, 2021). Las investigaciones con la perovskita como absorbente solar comenzaron a principios de 1990 (Tonui et al., 2018), pero solo en 2009 se demostró su utilidad (Kojima et al., 2009).

Las celdas solares de perovskita siguen los mismos principios de la unión *PN* que caracteriza el funcionamiento que las celdas de silicio, y por tanto se pueden caracterizar usando los mismos parámetros de medida (Jeong et al., 2021). Se caracterizan por una arquitectura multicapa, compuesta por materiales funcionales organizados de forma secuencial. Donde simplificando, la capa activa de perovskita se ubica entre dos capas selectivas de carga: la capa transportadora de electrones (*Electron Transport Layer*, ETL) y la transportadora de huecos (*Hole Transport Layer*, HTL). Y el dispositivo incorpora electrodos conductores en sus extremos, los cuales permiten la recolección de las cargas generadas, como se ilustra en la Figura 3.

Figura 3

Representación gráfica de la estructura y las capas contenidas en una celda solar de Perovskita.



2.1 Caracterización de celdas solares de Perovskita

Presentan propiedades opto-electrónicas que las hacen altamente atractivas para aplicaciones fotovoltaicas. Entre estas destacan su alto coeficiente de absorción, lo que permite una captación eficiente de la radiación incidente a diferentes longitudes de onda; un amplio rango de absorción de radiación; una elevada movilidad de portadores de carga, que favorece el transporte eficiente de electrones y huecos; y una baja energía de enlace excitónico, la cual contribuye a una adecuada separación de cargas dentro del material (Jiménez Ross, 2017). Además, los parámetros de desempeño permiten determinar rendimiento de las celdas (C.B.Honsberg and S.G.Bowden, 2019):

- *PCE* (eficiencia de conversión de Potencia), Reflejar el rendimiento de la propia celda solar, la eficiencia depende del espectro y la intensidad de la luz solar incidente y de la temperatura de la celda solar.

- V_{OC} (Voltaje de circuito abierto), es el voltaje máximo disponible generado por la celda solar, y esto ocurre con corriente cero.

- J_{SC} (Corriente de corto circuito), es la corriente de la celda solar cuando la celda está en cortocircuito.

- FF (factor de llenado), mide la relación entre la potencia máxima de la celda solar y el producto de Voc y Jsc.

2.2 Fabricación de celdas solares de Perovskita

Las celdas solares de Perovskita ha sido fabricadas para comercialización desde 2021, donde SAULE TECHNOLOGIES, una empresa con sede en Polonia, lanzó la primera línea de producción del mundo de paneles solares basados en tecnología perovskita (Shah et al., 2023). Debido al importante papel que tiene el método de fabricación de la Perovskita en las características de desempeño. Así la calidad de las capas, su uniformidad y cristalización son la meta a alcanzar en los diferentes métodos de fabricación, entre los más usados están (Sun et al., 2025):

- *Spin-coating* (El recubrimiento de centrifugado) consiste en depositar una pequeña cantidad de solución precursora sobre la superficie del sustrato, el cual posteriormente gira a alta velocidad. La fuerza centrífuga distribuye uniformemente la solución y favorece la formación de una película delgada de perovskita tras la evaporación del solvente y el proceso de cristalización (Sun et al., 2025).

- *Evaporation* (Evaporación), consiste en calentar los materiales precursores, generalmente mediante un sistema de calentamiento eléctrico sobre un crisol metálico, dentro de una cámara de vacío, hasta producir su evaporación. El vapor se deposita posteriormente sobre el sustrato, formando una película delgada de espesor controlado (Thankappan et al., 2025).

- *Sputtering* (Pulverización catódica), es una técnica de deposición física de vapor (PVD) en la que se genera un plasma, generalmente de argón, cuyos iones bombardean un material blanco (*target*). Como resultado del bombardeo, se desprenden átomos del material, los cuales se depositan posteriormente sobre el sustrato para formar una película delgada (Vilcot et al., 2017).

- *Spray Pyrolysis* (Pirólisis por pulverización), usando una boquilla la solución de perovskita se atomiza en pequeñas gotas que se esparcen por el sustrato calentado, formando la capa (Tahir et al., 2020).

- *Screen printing* (Serigrafía)consiste en transferir una tinta precursora sobre el sustrato mediante una pantalla o malla con un patrón definido. La tinta atraviesa las zonas abiertas de la pantalla y se deposita sobre el sustrato, reproduciendo el patrón deseado para la formación de la película (Sun et al., 2025).

- *Blade coating* (Recubrimiento de Cuchillas), consiste en extender una solución o tinta precursora sobre el sustrato mediante el desplazamiento de una cuchilla (*doctor blade*), obteniendo una película húmeda de espesor controlado. Posteriormente, el solvente se evapora y

la película se somete a procesos de secado y/o tratamiento térmico para favorecer su formación y cristalización (Sun et al., 2025).

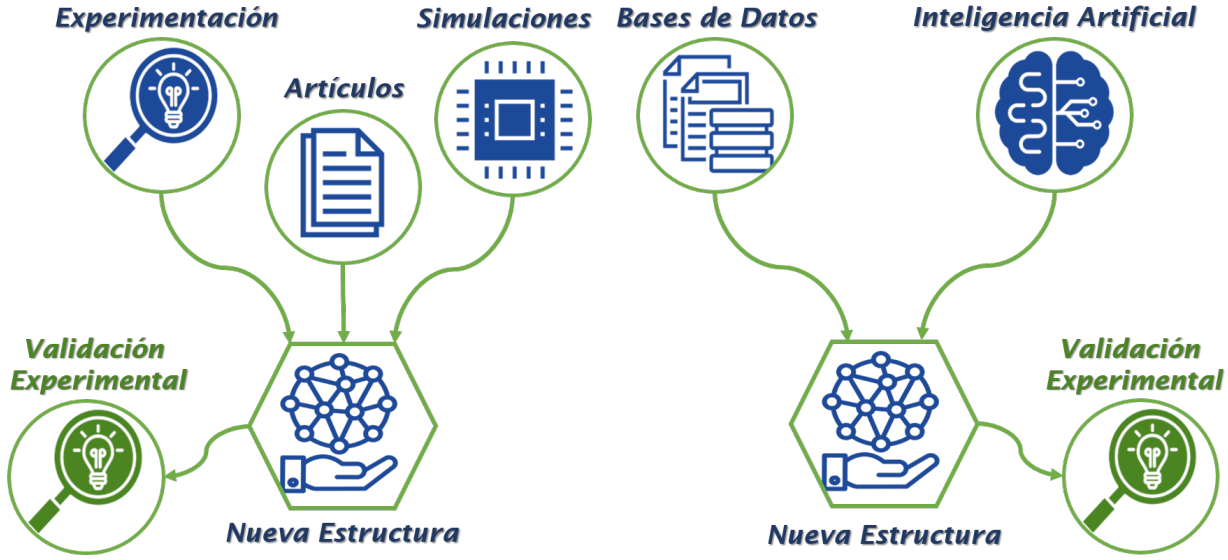
Luego, en estos métodos puede o no, que se aplique un recocido posterior de la capa.

2.3 La ciencia computacional en el descubrimiento de materiales

Tradicionalmente los científicos descubren nuevos materiales mediante investigación experimental, teórica o computacional convencional, donde los datos son los recursos y el conocimiento es el objetivo a conseguir. En particular durante el desarrollo de nuevos materiales, la cantidad de variables involucradas puede ser demasiado grande, y/o las relaciones entre estas variables puede ser demasiado complejo como para ser abordado utilizando procedimientos convencionales, lo que hace a este proceso lento, costoso e ineficiente (ver esquema de la izquierda en la Figura 4) (Himanen et al., 2019). Así nace los términos de *Data-driven materials science* (ciencia de materiales basada en datos) ó *Literature-based Discovery* (Descubrimiento basado en la literatura) (ver esquema de la derecha en la Figura 4) y *materials informatics* (informática de materiales) como el nuevo enfoque que hace uso de la combinación de conjuntos de datos, el aprendizaje automático (*Machine Learning*, ML) y métodos de cómputo de alto rendimiento para extraer sistemáticamente conocimiento de conjuntos de datos de materiales, lo resulta en una notable ayuda en el área de investigación de materiales (Himanen et al., 2019).

Figura 4

Esquema del enfoque tecnológico para el descubrimiento de materiales y/o estructuras.



Nota. Comparación entre el enfoque tradicional (izquierda) y el enfoque de *data-driven materials science* (ciencia de materiales basada en datos) (derecha). Adaptado de "Data-Driven Materials Science: Status, Challenges, and Perspectives" de Himanen et al. (2019).

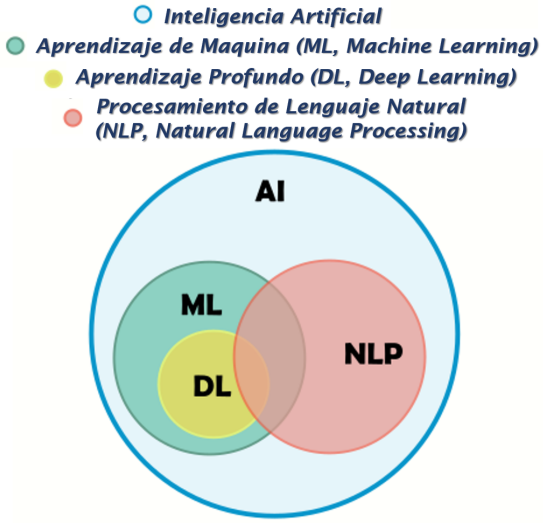
3 Procesamiento de Lenguaje Natural para La extracción de información

3.1 Procesamiento de Lenguaje Natural (Natural Language Processing, NLP)

El NLP es una subárea de la inteligencia artificial (Figura 5) encargada de analizar, comprender y procesar la información comunicada, necesaria para el análisis y simplificación de texto, en especial en la extracción de información (*Information Exraction*, IE) (Kang et al., 2020).

Figura 5

Diagrama de la distribución general del procesamiento natural del lenguaje (NLP) como parte de la inteligencia artificial.



Nota. Adaptado de "AI, Machine Learning (ML) and Natural Language Processing (NLP)" de AthenaTech LLC (2019).

El NLP abarca un conjunto amplio de tareas orientadas al análisis y comprensión del lenguaje humano, incluyendo el análisis sintáctico, el etiquetado de partes del discurso (POS), la traducción automática y el desarrollo de sistemas de diálogo. Esto a dado lugar al desarrollo de

dispositivos inteligentes tales como, traducciones simultáneas (*Siri*, *Cortana*), reconocimiento de voz (*iflyrec 3* de iFLYTEK), predicción de mercados mediante análisis de sentimientos usando asistentes robóticos de servicio al cliente (*Alibaba 5*), etc (Kang et al., 2020). También de asistentes IA (inteligencia artificial) como *ChatGPT* de OpenAI (OpenAI, 2022) o *Copilot* de Microft (Microsoft, 2023). Así mismo, en el ámbito de ciencia y la investigación científica, los académicos están usando el NLP para ayudar en el descubrimiento de nuevos materiales.

3.2 Extracción de información (Information Extraction, IE)

La Extracción de información es la estrategia actualmente usada para la recuperación de información a partir de textos. Se encarga de aislar los fragmentos de texto relevantes, extraer la información pertinente de estos fragmentos para luego convertirlos en información estructurada. El objetivo de la investigación en IE es desarrollar sistemas que encuentren y vinculen la información relevante, ignorando la información superflua e irrelevante (Hobbs and Riloff, 2010).

Los métodos para aplicar la IE se pueden dividir en, basadas en la Ingeniería del Conocimiento (*Knowledge Engineering*, KE) y métodos basados en *Machine learning*. Lo métodos KE son enfoques basados en reglas heurísticas y conocimientos, que permiten integrar los conocimientos humanos a sistemas informáticos. Por su parte, los enfoques basados en ML emplean distintos tipos de aprendizaje, tales como el aprendizaje supervisado (*Supervised Learning*), semi-supervisado (*Semi-Supervised Learning*), no supervisado (*Unsupervised Learning*) y auto-supervisado (*Self-Supervised Learning*), los cuales se diferencian principalmente por el tipo de información disponible durante el entrenamiento (Mannai et al., 2017; Golshan et al., 2018; Amini et al., 2024); y distintos tipos de estrategias de entrenamiento y adaptación, entre las que destacan

el Entrenamiento desde cero (*Training from Scratch*), el Aprendizaje por Transferencia (*transfer learning*), el Aprendizaje Multitarea (*multitask learning*) y el Aprendizaje Activo (*active learning*) (Zhang and Yang, 2017; Schröder and Niekler, 2020; Bharadiya, 2023). Estas estrategias y tipos serán abordados con mayor detalle en la Secciones 3.6 y 3.7.

Entre los modelos ML más utilizados para aplicaciones NLP están, los modelos convencionales como, los los Árboles de Decisión (DT) y EL Campo Aleatorio Condicional (CRF), etc.Y Los modelos de aprendizaje profundo (*Deep Learning*, DL) como, las Redes Neuronales Convolucionales (CNN), las Redes Neuronales Recurrentes (RNN) y los Modelos de Lenguaje Preentrenados (*Pre-trained Language Models*, PLMs), etc (Golshan et al., 2018).

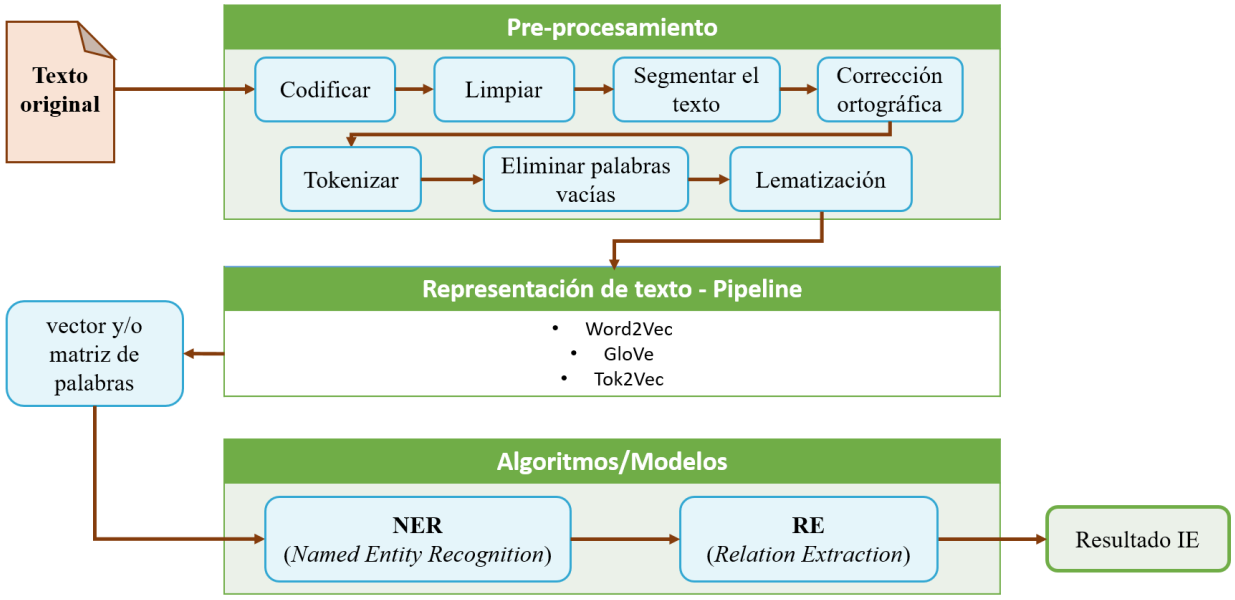
Las diversas sub-tareas involucradas en la IE son: Enlace de Entidades Nombradas (NEL), Resolución de Correferencias (CR), Extracción de Información Temporal, Extracción de eventos (EE), Construcción de Bases de Conocimiento y Razonamiento y las dos más usadas en el NLP, el Reconocimiento de Entidades Nombradas (*Named Entity Recognition*, NER) y la Extracción de Relaciones (*Relation Extraction*, RE) (Piskorski and Yangarber, 2012; Singh, 2018).

3.3 Arquitectura de la IE

Aunque los sistemas de IE construidos para diferentes tareas pueden diferir significativamente entre sí, de forma general, para desarrollar un sistema NLP centrado en las tareas de extracción de entidades nombradas (NER) y la extracción de relaciones (RE), el proceso puede incluir: el preprocesamiento del texto, su representación y, finalmente, el entrenamiento y aplicación de modelos mediante algoritmos adecuados para la tarea específica, como se ilustra en la Figura 6.

Figura 6

Diagrama general de la arquitectura de la extracción de información, IE.



Nota. Adaptado de los artículos: Singh (2018) y Kang et al. (2020) .

El preprocesamiento tiene como objetivo limpiar y normalizar el texto, con el fin de mejorar la eficiencia y precisión en las etapas posteriores. Este proceso incluye tareas como la corrección ortográfica, la tokenización, que consiste en segmentar el texto en unidades básicas; el etiquetado de partes del discurso (POS), que asigna categorías gramaticales a cada token; la eliminación de palabras vacías, orientada a reducir información irrelevante; y la lematización, que permite normalizar las palabras a su forma base. Estas etapas facilitan la preparación del texto para tareas posteriores como NER. En enfoques modernos basados en modelos de *Deep Learning*, algunas de estas etapas pueden ser implícitamente aprendidas por el modelo.

3.4 Modelos de Ingeniería del Conocimiento Knowledge Engineering, KE)

Los modelos KE utilizan reglas heurísticas y conocimientos para integrar los conocimientos humanos a sistemas informáticos, fueron los primeros modelos usados para el NLP (Manning and Schutze, 1999; Jurafsky and Martin, 2026). Estas reglas suelen ser reglas heurísticas como, patrones lingüísticos, expresiones regulares o estar basadas en conocimiento especifico de un dominio, permitiendo identificar estructuras gramaticales dentro del texto. Suelen ser más fáciles de interpretar y controlar dado que estas reglas controlan completamente el sistema, pero dependiendo del área llegan a ser complicadas de implementar. Esto hace que tengan una baja capacidad de generalización y escalado (Manning and Schutze, 1999; Goldberg, 2017).

Actualmente estos KE han sido reemplazados casi por completo por los modelos ML y *Deep Learning* (Goldberg, 2017; Lauriola et al., 2022). No obstante, el uso de reglas heurísticas como las del KE aun son utilizados como mecanismos de apoyo en modelos NLP, particularmente en etapas de preprocesamiento o post-procesamiento.

3.5 Modelos de Aprendizaje Profundo (Deep Learning, DL)

El *Machine Learning* (ML) comprende un conjunto amplio de métodos para el análisis de datos y se pueden dividir en dos, los algoritmos/modelos del ML convencional y los de *Deep Learning* (DL). En los enfoques tradicionales de Machine Learning, el desempeño del modelo suele depender en gran medida de la selección y construcción manual de características a partir del texto preprocesado (tokeinzacion, lematizacion, etc). Por el contrario, los modelos de Deep Learning son capaces de aprender automáticamente representaciones distribuidas de los datos,

reduciendo la necesidad de ingeniería manual de características, aunque continúan requiriendo etapas cortas de preprocesamiento y conjuntos de datos adecuados para su entrenamiento.

Se definen como modelos de *Deep Learning* a todos los modelos *Machine Learning* (ML) basados en redes neuronales de múltiples capas, capaces de aprender representaciones jerárquicas de los datos. Las redes neuronales son una representación artificial de una neurona biológica que funciona mediante la aplicación sucesiva de funciones simples, generalmente no lineales. Donde cada una de estas funciones recibe múltiples entradas y pesos asociados a cada entrada, y produce una salida que sirve como entrada para las capas posteriores. De esta manera, el modelo construye progresivamente representaciones más complejas a partir de los datos iniciales. (Kamath et al., 2019). Debido a la evolución de los modelos ML, algunos trabajos clasifican las redes neuronales más simples dentro del *Machine Learning* convencional, diferenciándolas de los modelos *Deep Learning* (Chauhan and Singh, 2018).

3.6 Tipos de aprendizaje de entrenamiento y adaptación de Modelos

Para el entrenamiento y adaptación de modelos basados en *Deep Learning* se emplean distintos tipos de aprendizaje, los supervisados, no-supervisados, semisupervisados y auto-supervisados. En NLP, los enfoques supervisados son ampliamente utilizados debido a la disponibilidad de corpus anotados y constituyen la base de los modelos empleados en este trabajo. La Tabla 1 resume de la principal característica, las ventajas y desventajas de cada tipo de aprendizaje.

Tabla 1

Principales tipos de aprendizaje utilizados en ML/DL.

Tipo	**Ventaja principal**	**Limitación principal**
Supervisado	Alta precisión cuando se dispone de datos etiquetados.	Requiere grandes cantidades de datos anotados.
No supervisado	Permite descubrir patrones sin necesidad de etiquetas.	Los resultados son más difíciles de evaluar e interpretar.
Semi-supervisado	Reduce el esfuerzo de anotación combinando datos etiquetados y no etiquetados.	Mayor complejidad de implementación.
Auto-supervisado	Aprovecha grandes volúmenes de datos no etiquetados para preentrenamiento.	Alto costo computacional y dependencia de la tarea de pretexto.

3.7 Estrategias de entrenamiento y adaptación de Modelos

Como se introdujo previamente en la Sección 3.2 para aplicar la IE es posible usar diferentes estrategias de entrenamiento y adaptación para los modelos de ML y/o *Deep Learning*. En la Tabla 2 se resumen las principales ventajas y desventajas de las estrategias entrenamiento y adaptación, donde se observa que el *Transfer Learning* ofrece una combinación óptima de eficiencia, simplicidad de implementación y rendimiento.

Tabla 2

Estrategias de entrenamiento y adaptación de modelos de ML/DL.

Estrategia	Ventajas	Limitaciones
Transfer Learning	- Reduce la necesidad de datos etiquetados. - Aprovecha modelos preentrenados y acelera el entrenamiento.	- Requiere ajuste fino adecuado. - Su desempeño depende de la similitud entre dominios.
Training from Scratch	- Máxima adaptación al dominio y control del modelo.	- Requiere grandes volúmenes de datos y recursos computacionales. - Generalmente implica mayores tiempos de entrenamiento.
Multitask Learning	- Mejora la generalización mediante el aprendizaje compartido entre tareas.	- Requiere diseñar tareas relacionadas y datos adecuados. - Puede presentar interferencia entre tareas.
Active Learning	- Reduce el esfuerzo de anotación al seleccionar muestras informativas.	- Requiere ciclos iterativos y participación humana continua. - Puede introducir sesgos de selección.

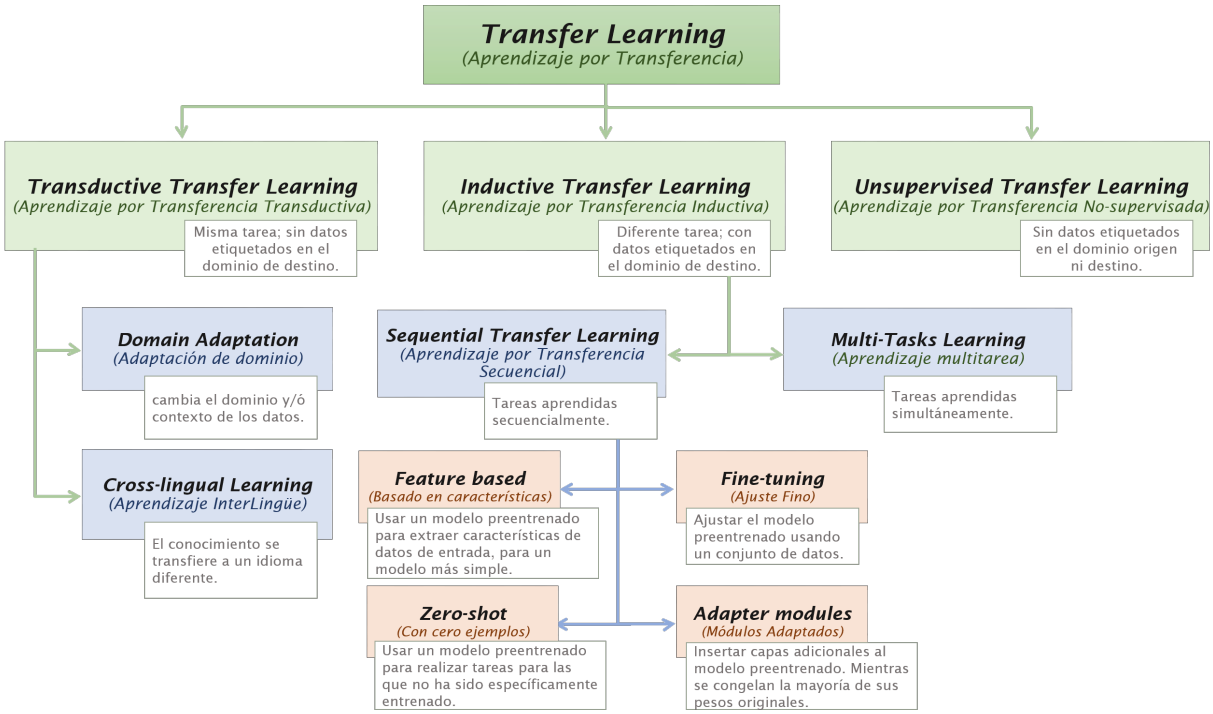
3.8 Técnicas del Aprendizaje por transferencia (Transfer Learning)

El Aprendizaje por transferencia (*Transfer Learning*) hace uso de un modelo ya entrenado o preentrenado en una disciplina o tarea suficientemente parecida, como base, al cual se aplica un nuevo entrenamiento más especifico, una adaptación o un afinamiento, para obtener un nuevo modelo más limpio y preciso en su tarea, respecto a el modelo base (Alzubaidi et al., 2023; Sabha et al., 2024).

Puede aplicarse tanto en modelos de *Machine Learning* convencionales como a modelos *Deep Learning*. Sin embargo, su uso es particularmente efectivo en los modelos de lenguaje preentrenados (*Pre-trained Language Models*, PLMs) descritos en la Sección 3.9, donde la arquitectura *Transformer* permite reutilizar representaciones aprendidas previamente, reduciendo la necesidad de entrenar modelos desde cero y facilitando su aplicación en escenarios con cantidades limitadas de datos etiquetados (Bharadiya, 2023).

Figura 7

Ilustración de los métodos de Transfer Learning más comúnmente aplicados en el NLP.



Nota. Adaptado de la información contenida en los artículos: Pan and Yang (2010), Ruder (2019) y Alyafeai et al. (2020).

Para aplicar el *Transfer Learning* en el area del NLP existen diferentes técnicas y enfoques. Según los artículos (Pan and Yang, 2010; Ruder, 2019; Alyafeai et al., 2020; Zhao et al., 2024) las técnicas para la aplicación del Transfer Learning se pueden categorizar en base a los atributos de los datos y las tareas, en tres tipos principales: el Aprendizaje por Transferencia Transductiva (*Transductive Transfer Learning*), el Aprendizaje por Transferencia Inducitiva (*Inductive Transfer Learning*) y el Aprendizaje por Transferencia No supervisado (*Unsupervised Transfer Learning*). También existe un denominado Aprendizaje por Transferencia Heterogénea (*Heterogeneous Transfer Learning*) donde se combinan los demás tipos dependiendo la necesidad. En la Figura 7 se muestra un mapa conceptual que ilustra los tres tipos de *Transfer Learning*.

El proceso de aplicación del *Transfer Learning* depende del enfoque y la situación especifica en el que se va a usar; donde, las fases pueden o no ser claramente separadas y adaptadas para satisfacer necesidades específicas de la tarea. El enfoque más utilizado y conocido es el *Sequential Transfer Learning*. Este se puede dividir en dos fases (Ruder, 2019):

1. *Pre-entrenamiento:* El entrenamiento previo en la tarea fuente donde es necesario contar con una gran cantidad de datos. Debido al alto costo computacional del entrenamiento desde cero, actualmente es común saltarse este paso al reutilizar modelos previamente entrenados, en especial los obtenidos a partir de grandes volúmenes de datos no etiquetados (Trewartha et al., 2022; Wang et al., 2023). Entre los modelos previamente entrenados más usados se encuentran los diversos Modelos de Lenguaje Preentrenados (PLMs).

2. *Adaptación / Aplicación del Transfer learning:* En esta fase como su nombre lo indica se aplica el método deseado de *transfer learning* sobre el modelo preentrenado: *fine-tuning*, *Adapter modules*, *Feature based*, *Zero-shot* o una combinación de métodos (*Heterogeneous Transfer Learning*), ver la Figura 7.

3.8.1 Ajuste Fino (Fine-Tuning)

Dentro de las estrategias de *Transfer Learning*, uno de los enfoques más utilizados es el *Fine-Tuning*, donde un modelo previamente entrenado es ajustado para una tarea específica mediante entrenamiento adicional. A diferencia del por ejemplo del enfoque *Feature-Based*, donde el modelo actúa únicamente como extractor de características y sus parámetros permanecen congelados, en el *Fine-Tuning* los parámetros internos del modelo son actualizados durante el entrenamiento, permitiendo adaptar sus representaciones al nuevo problema. Estos parámetros pueden ser actualizados en todas las capas o solo en algunas capas (Ruder, 2019; Alyafeai et al., 2020).

El *Fine-Tuning* es usado en modelos PLMs como BERT, un codificador Transformer bidireccional de múltiples capas basado en la implementación original de (Vaswani et al., 2017), donde esta estrategia es usada para garantiza que se mantengan las capacidades generales de comprensión del modelo mientras se adapta a aplicaciones específicas de un dominio luego del preentrenamiento. Permitiendo que el modelo destaque en tareas especializadas sin sacrificar su amplio conocimiento (Devlin et al., 2019; Wu et al., 2025).

Entre las principales ventajas del *Fine-Tuning* se encuentran la reducción en los requerimientos de datos etiquetados y la posibilidad de aprovechar conocimiento lingüístico previamente aprendido. Sin embargo, este enfoque puede presentar desafíos relacionados con el costo compu-

tacional y la selección adecuada de “Hiperparámetros”, especialmente si el modelo tiene un gran tamaño (Wu et al., 2025).

3.9 Modelos de Lenguaje Preentrenados (Pre-trained Language Models, PLMs)

Los Modelos de Lenguaje Preentrenados (*Pre-trained Language Models*, PLMs) son modelos basados principalmente en la arquitectura *Transformer*, la cual representa una evolución respecto a enfoques tradicionales de *Deep Learning* (DL) como las Redes Neuronales Convolucionales (CNN) y las Redes Neuronales Recurrentes (RNN). Introducida por Vaswani et al. (Vaswani et al., 2017), esta arquitectura emplea mecanismos de auto-atención que permiten identificar las relaciones contextuales entre las palabras de una secuencia y asignarles diferentes niveles de importancia. A través de múltiples capas de procesamiento, los *Transformers* generan representaciones contextualizadas del texto, permitiendo capturar dependencias complejas y mejorar el desempeño en diversas tareas de NLP, respecto a los modelos convencionales de DL (Patwardhan et al., 2023; Yao and Yuan, 2024). Los modelos basados en esta arquitectura Transformer han marcado un punto de inflexión en NLP y se pueden dividir en, modelos preentrenados de conocimiento general, como (BERT) (Chakkarwar et al., 2023) o (SciBERT) (Beltagy et al., 2019), y los más actuales Modelos de Lenguaje a Gran Escala (*Large Language Models*, LLMs) como GPT-3 o ChatGPT (OpenAI, 2022), que amplían la escala y capacidades hacia la generación de texto y diálogo.

Los PLMs se obtienen inicialmente mediante entrenamiento desde cero (*Training from Scratch*), a través de un proceso de preentrenamiento sobre grandes volúmenes de datos generales. Posteriormente, estos modelos pueden ser adaptados a tareas específicas mediante estrategias de *Transfer Learning*, entre las cuales, el Ajuste Fino (*fine-tuning*) es una de las más comunes.

Sin embargo, el ajuste fino de los Modelos de Lenguaje Preentrenados (PLMs), especialmente de aquellos de gran tamaño como el caso de los LLMs, continúa siendo un desafío debido a los elevados requerimientos computacionales y a la limitada disponibilidad de datos etiquetados en tareas específicas (Tinn et al., 2023; Li et al., 2025). En este contexto, el uso de PLMs más compactos constituye una alternativa viable, que puede alcanzar un rendimiento comparable al de los LLMs, ofreciendo además ventajas en términos de eficiencia computacional y facilidad de implementación en determinadas aplicaciones (Patwardhan et al., 2023).

3.10 Reconocimiento de entidades nombradas (Named Entity Recognition, NER)

La tarea llamada "Reconocimiento de entidades nombradas (*Named Entity Recognition*, NER)" tiene como objetivo identificar y clasificar entidades relevantes dentro de un texto en categorías predefinidas, tales como ubicaciones, organizaciones o personas (Piskorski and Yangarber, 2012). Para cada segmento de texto, puede ser una sola palabra o varias palabras con categorías correspondientes. En la Figura 8 se ilustra un ejemplo del resultado esperado de la tarea NER.

En el desarrollo de modelos para el reconocimiento de entidades nombradas (NER) se distinguen principalmente tres enfoques: los basados en reglas construidas a mano, basados en ML y/o DL, y el híbrido que combina elementos de ambos.

Los modelos *Deep Learning* (DL) actuales funcionan notablemente mejor que las técnicas convencionales ML a la hora de realizar tarea NER, permitiendo la simplificación del trabajo de preprocesamiento y de representación del texto, absorbiendo la mayor parte del trabajo de tokenización, POS, la eliminación de palabras vacías, la lematizacion y la generación del vector y/o

matriz de palabras (Hu et al., 2024b). Y si los aplicamos como sistemas híbridos aumentamos la precisión y flexibilidad en comparación (Naseer et al., 2021).

Figura 8

Ejemplo de la aplicación de la tarea NER.

…. mesoporous TiO2 ETL / CH3NH3PbI3 PTF /HTMs/Au solar cells. The pyrene-based derivative Py-C exhibited a short-circuit current density JSC of 20.2 mA/cm2 NJSC, an open-circuit voltage VOC (Voc) of 0.886 V NVOC, and a fill factor FF of 69.4% NFF under an illumination of 1 sun (100 mW/cm2), resulting in an overall power conversion efficiency PCE of 12.4% NPCE. The performance is comparable to that of the well-studied spiro-OMeTAD HTL, even though the Voc is slightly lower….

Entidad	HTL	ETL	PTF
valor	spiro-OMeTAD	TiO2	CH3NH3PbI3

Entidad	JSC	VOC	FF	PCE
valor	short-circuit current density	open-circuit voltage	fill factor	conversion efficiency

Entidad	NJSC	NVOC	NFF	NPCE
valor	20.2 mA/cm2	0.886 V	69.4 %	12.4 %

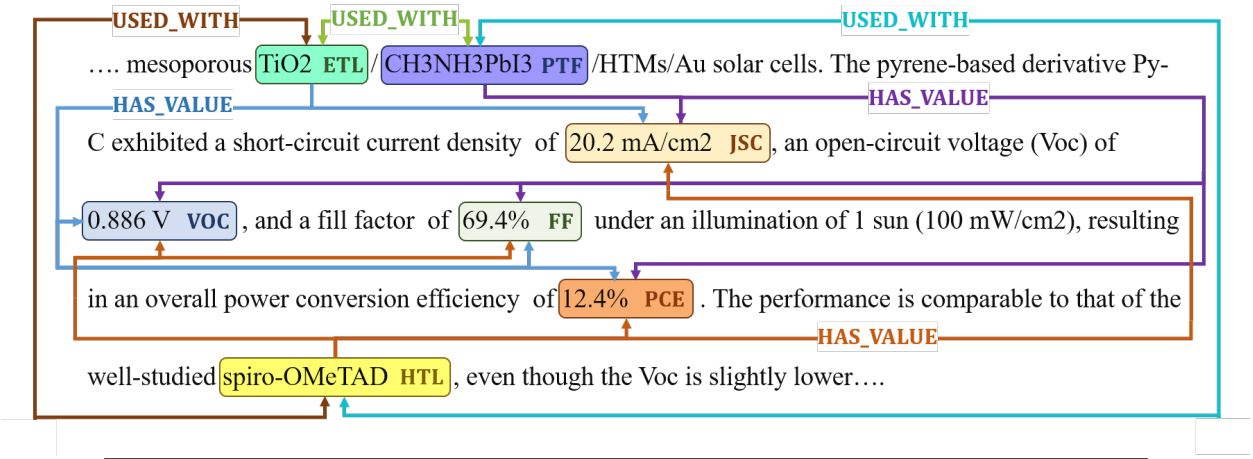
Nota. Ilustra la salida esperada de la extracción de entidades nombradas, usando de una parte de la sección del resumen (Abstract) del artículo "Efficient inorganic–organic hybrid perovskite solar cells based on pyrene arylamine derivatives as hole-transporting materials" de Jeon et al. (2013).

3.11 Extracción de relaciones (Relation Extraction, RE)

La extracción de relaciones (*Relation Extraction*, RE) es la tarea de detectar y clasificar relaciones entre entidades identificadas en un texto, creando pares entidad 1 y 2 - relación. En la Figura 9 se ilustra un ejemplo del resultado esperado de la tarea RE.

Figura 9

Ejemplo de la aplicación de la tarea RE.



Entidad1	Entidad2	Relacion	Definición de la relación
TiO2	CH3NH3PbI3	USED_WITH	Relación estructural
TiO2	spiro-OMeTAD	USED_WITH	Relación estructural
CH3NH3PbI3	TiO2	USED_WITH	Relación estructural
TiO2	20.2 mA/cm2	HAS_VALUE	Relación de desempeño
TiO2	0.886 V	HAS_VALUE	Relación de desempeño
TiO2	69.4 %	HAS_VALUE	Relación de desempeño
TiO2	12.4 %	HAS_VALUE	Relación de desempeño
CH3NH3PbI3	20.2 mA/cm2	HAS_VALUE	Relación de desempeño
CH3NH3PbI3	0.886 V	HAS_VALUE	Relación de desempeño
CH3NH3PbI3	69.4 %	HAS_VALUE	Relación de desempeño
CH3NH3PbI3	12.4 %	HAS_VALUE	Relación de desempeño
spiro-OMeTAD	20.2 mA/cm2	HAS_VALUE	Relación de desempeño
spiro-OMeTAD	0.886 V	HAS_VALUE	Relación de desempeño
spiro-OMeTAD	69.4 %	HAS_VALUE	Relación de desempeño
spiro-OMeTAD	12.4 %	HAS_VALUE	Relación de desempeño

Nota. Ilustra los pares entidad-relación esperados a partir de una parte de la sección del resumen (Abstract) del artículo “Efficient inorganic–organic hybrid perovskite solar cells based on pyrene arylamine derivatives as hole-transporting materials” de Jeon et al. (2013).

De manera similar al NER, los enfoques para resolver la tarea RE se pueden dividir en tres: métodos de patrones construidos a mano, métodos de ML y/o DL subdivididos en semisupervisados, supervisados y no supervisados, y los métodos de supervisión a distancia (Yang et al., 2022).

Los enfoques basados en *Deep Learning* (DL) reducen la dependencia de características lingüísticas diseñadas manualmente y favorecen la transferencia entre dominios. En los métodos no-supervisados y semisupervisados, generalmente no se requieren clases de relaciones predefinidas ni grandes cantidades de datos anotados, ya que las relaciones se identifican a partir de patrones presentes en los datos. Y en los enfoques supervisados se suelen formular la RE como un problema de clasificación multiclase, donde un modelo entrenado con datos etiquetados predice la categoría a la que pertenece un par de entidades (Kumar, 2017; Yang et al., 2022).

La tarea RE se puede hacer mediante dos enfoques, basados en un *Pipeline* (tubería) o la extracción conjunta (*Joint Extraction*). El método basado en una *Pipeline* es más sencillo y flexible, abordando la extracción de entidades y relaciones de manera secuencial, donde por ejemplo, se identifican las entidades mediante técnicas de NER y posteriormente se utilizan dichas entidades como entrada para el modelo de RE. Esto puede acarrear la propagación de errores. Por el contrario, el *Joint Extraction* modela ambas tareas de manera integrada, utilizando un único modelo, que permite capturar dependencias entre ambas tareas, pero necesita una implementación más compleja (Yang et al., 2022; Wang et al., 2022). En la Tabla 3 se ve un resumen de las ventajas y desventajas de los dos enfoques.

Tabla 3

Resumen de las ventajas y desventajas de los dos enfoques para aplicar la tarea RE.

Método	**Ventajas**	**Desventajas**
Pipeline	- Sistemas modulares. - Fácil de implementar. - Permite la reutilización de modelos.	- Permite la propagación de errores. - No hay dependencia entre tareas.
Joint Extraction	- Mejor dependencia contextual. - Menos propagación de errores	- Son más complejos. - Requieren más datos. - El entrenamiento es más difícil.

Aunque la mayoría de enfoques de extracción de relaciones se han centrado en el análisis a nivel de oración, debido a la complejidad de modelar dependencias entre múltiples oraciones, existen aproximaciones que buscan incorporar contexto más amplio, como el uso de ventanas de oraciones o enfoques a nivel de documento. La tarea RE a nivel de documento es un campo activo de investigación, donde los modelos tienden a perder fiabilidad o ser menos estables al manejar tareas de multi-oración directamente (Wang et al., 2022; Diaz-Garcia and Lopez, 2025).

3.12 Desarrollo de modelos NLP

En la red existen librerías que permiten el desarrollo de modelos de ML y DL para tareas de NLP, como *NLTK* (Loper and Bird, 2002), *spaCy* (Honnibal et al., 2015), *CoreNLP* (Manning et al., 2014), *AllenNLP* (Gardner et al., 2017), *Vertex AI* (Google Cloud, 2021) y *HuggingFace's Transformers* (Wolf et al., 2019), etc. Estas herramientas y/o bibliotecas tienen variados enfoques, desarrollo y adaptabilidad. Muchas de estas han sido implementadas en Python debido a su amplia

adopción en la comunidad científica. La Tabla 4 presenta un resumen de algunas herramientas y/o bibliotecas disponibles en la red.

Tabla 4

Principales características y detalles de las principales herramientas, bibliotecas y modelos de NLP.

Librería	Lanzamiento	Licencia
NLTK[1] Natural Language Toolkit, es una plataforma para trabajar con datos de lenguaje humano.	2001	Apache 2.0
Gensim[2] Es una biblioteca para modelado de temas, indexación de documentos y recuperación de similitudes.	2009	GNU LGPLv2+
SpaCy[3] Es una biblioteca industrial para realizar tareas de NLP en Python.	2015	MIT license
AllenNLP[4] Plataforma para resolver tareas NLP en PyTorch.	2019	Apache 2.0
HuggingFace's Transformers[5] Librería y plataforma que proporciona arquitecturas de propósito general (BERT, GPT-2, RoBERTa, XLNet, etc).	2019	Apache 2.0
CoreNLP[6] Es una biblioteca de PNL que abarca los principales módulos y tareas de PNL, es decir, tokenización, etiquetado PoS, análisis, NER.	2010	GNU GPLv3
Stanza[7] Es una colección de herramientas precisas y eficientes para muchos lenguajes humanos.	2019	Apache 2.0

[1] Disponible:`https://www.nltk.org/`
[2] Disponible:`https://radimrehurek.com/gensim/`
[3] Disponible:`https://spacy.io/`
[4] Disponible:`https://allenai.org/allennlp/software/allennlp-library`
[5] Disponible:`https://huggingface.co/transformers/`
[6] Disponible:`https://stanfordnlp.github.io/CoreNLP`
[7] Disponible:`https://stanfordnlp.github.io/stanza/`

Nota. Adaptada de "An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools" de Lauriola et al. (2022).

Aunque las bibliotecas disponibles ofrecen diferentes funcionalidades y niveles de abstracción, el proceso general de construcción de un modelo de *Deep Learning* sigue una secuencia de etapas similares, independientemente del tipo de aprendizaje o la estrategia de entrenamiento:

1. Transformar lo datos de entrenamiento en representaciones numéricas adecuadas (por ejemplo, embeddings en NLP como Word2Vec), que permitan al modelo procesar la información textual.

2. Un entrenamiento que permite ajustar parámetros internos que corresponden normalmente a los valores de los pesos en las redes neuronales o bloques *transformers* del modelo.

3. La evaluación del modelo utilizando métricas específicas según la tarea, permitiendo medir su desempeño y capacidad de generalización.

3.12.1 Representación de los datos

Para poder procesar texto, un modelo ML y/o *Deep Learning* necesita antes, que el texto se transforme en representaciones numéricas que capturen información semántica y sintáctica. Para hacerlo, se representan las palabras mediante vectores densos conocidos como *embeddings*, los cuales capturan relaciones semánticas entre palabras. Uno de los enfoques más influyentes es Word2Vec, propuesto por (Mikolov et al., 2013), que aprende representaciones vectoriales a partir del contexto en el que aparecen las palabras. Los *embeddings* son tradicionalmente estáticos, donde cada palabra tiene una única representación independientemente del contexto (Goldberg, 2017).

Mientras librerías clásicas de ML como *scikit-learn* (Pedregosa et al., 2011) usan representaciones explícitas de los datos (por ejemplo, TF-IDF), herramientas como *spaCy* (Honnibal

et al., 2015) cuentan con mecanismos para generar *embeddings* integrados dentro del *pipeline* del modelo. Por otra parte, PLMs como BERT (Chakkarwar et al., 2023), generan representaciones contextuales, donde el significado de una palabra depende del contexto en el que aparece.

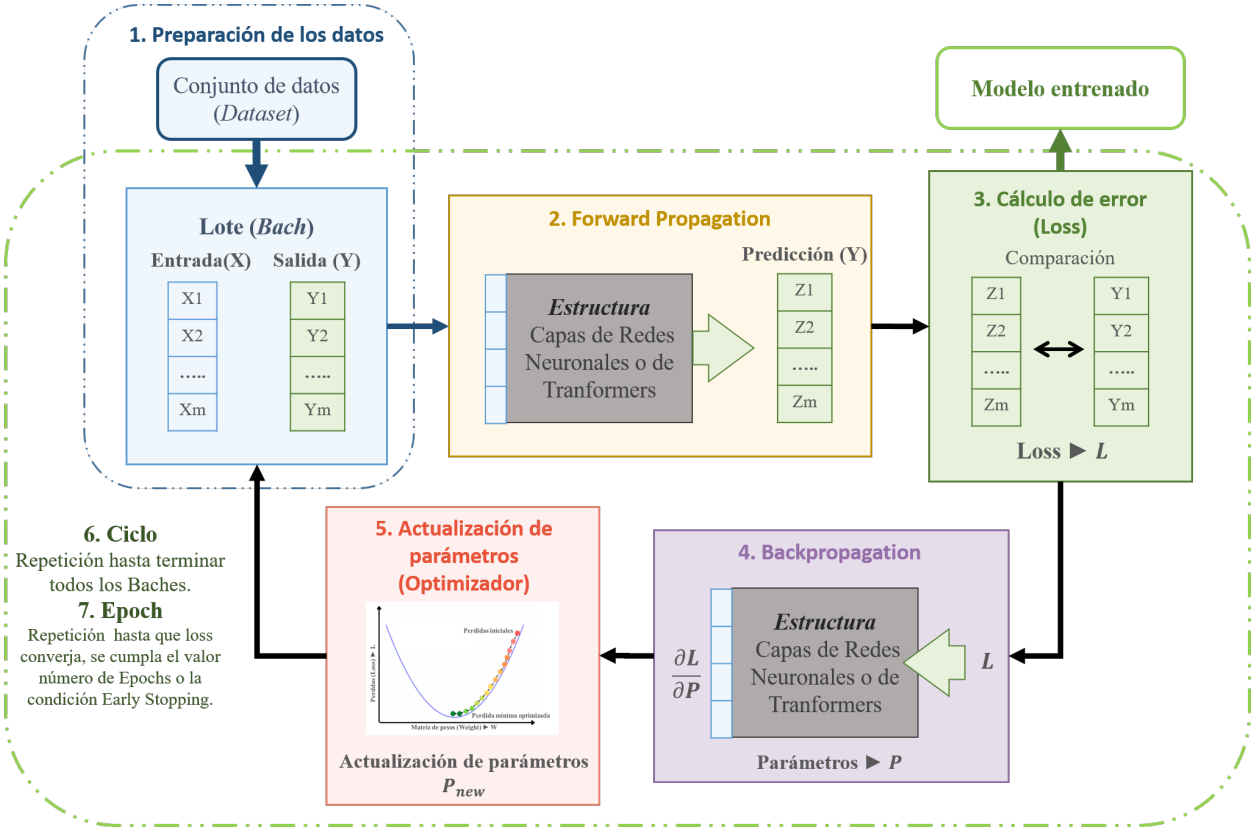
3.12.2 Entrenamiento

Para realizar el proceso de entrenamiento se requiere de la definición de "Hiperparámetros", configuraciones externas que controlan el comportamiento del entrenamiento y que son establecidos antes. Entre los más comunes están, la tasa de aprendizaje (*Learning Rate*), el número de épocas (*Epochs*) y el tamaño del lote (*Batches*), etc. También, existen hiperparámetros para aplicar técnicas de regularización como, la parada temprana (*Early Stopping*), la tasa de olvido (*Dropout*)) y regularizacion L2 (*Weight Decay*) (Bengio et al., 2017; Krohn et al., 2019).

Aunque el proceso de entrenamiento de modelos de *Deep Learning* puede variar según la estrategia empleada (Sección 3.7), los principios fundamentales mediante los cuales el modelo aprende se mantienen. Independientemente de la arquitectura utilizada, y particularmente en los PLMs, el entrenamiento consiste en optimizar los parámetros internos del modelo mediante un proceso iterativo que incluye la propagación hacia adelante para generar una predicción, el cálculo del error, la retropropagación y la actualización de parámetros (Bengio et al., 2017; Krohn et al., 2019). Este ciclo se repite hasta alcanzar el criterio de convergencia establecido. Este proceso se ilustran en la Figura 10.

Figura 10

Esquema general del proceso interno de aprendizaje de un modelo de aprendizaje profundo (Deep Learning, DL).



Nota. Adaptado de los artículos: Bengio et al. (2017) y Krohn et al. (2019).

3.12.3 Hiperparámetros de Entrenamiento

Los Hiperparámetros de entrenamiento más relevantes son el *learning rate*, el número de epocas (*epochs*), el tamaño de lote (*batch size*) y las técnicas de regularización.

3.12.4 Regularización

La regularización agrupa un conjunto de técnicas utilizadas durante el entrenamiento de modelos que buscan evitar el sobre-ajuste (*overfitting*), es decir, evitar que el modelo memorice el conjunto de datos (*Dataset*) en lugar de generalizar:

- *La Tasa de Olvido (Dropout)*, consiste en eliminar de forma aleatoria neuronas y conexiones relevantes de la red neuronal del modelo durante el entrenamiento a fin de tratar de aumentar la habilidad del modelo a responder adecuadamente ante datos nunca vistos (Antona Castañares, 2020; Baciero Fernández, 2020).

- *La Parada Temprana (Early Stopping)* es una técnica que detiene el entrenamiento cuando el desempeño sobre el conjunto de evalaución no mejora durante un número predefinido de épocas (patience) (Bengio et al., 2017).

- *Regularizacion L2 (Weight Decay)*, penaliza pesos grandes en la función de pérdida, es decir evita que ciertos parámetros dominen, asi el modelo aprende representaciones más equilibradas y generaliza mejor (Krohn et al., 2019).

3.12.5 Técnicas complementarias

Además de la regularización existen otras técnicas que se pueden aplicar durante el entrenamiento para el mejoramiento del modelo:

- *Balanceo de clases*. Utilizada para mitigar distribuciones desiguales entre etiquetas dentro del conjunto de entrenamiento. Su objetivo es reducir el sesgo hacia las clases mayorita-

rias mediante estrategias como ponderación de clases, sobremuestreo o submuestreo (Seow et al., 2025).

- *Ajuste progresivo del tamaño del lote (Batch)*. Estrategia utilizada para incrementar gradualmente el tamaño del lote durante el entrenamiento, favoreciendo la estabilidad del aprendizaje en las etapas iniciales y una mayor eficiencia computacional posteriormente. En spaCy puede implementarse mediante funciones de crecimiento progresivo como $compounding(4.0, 32.0, 1.001)$ (Honnibal et al., 2015).

3.12.6 Evaluación

La evaluación se aplica para verificar la precisión de las herramientas y modelos para realizar su tarea. Las métricas del análisis estadístico de la evaluación de modelos NER y RE son (Murphy, 2012) (Eisenstein, 2019):

TP : Número de entidades/Relaciones correctamente identificadas como positivas
TN : Número de entidades/Relaciones correctamente identificadas como negativas
FP : Entidades/Relaciones negativas incorrectamente etiquetadas como positivas
FN : Entidades/Relaciones positivas incorrectamente etiquetadas como negativas

- **Precisión (Precision)**, permite medir la calidad del modelo en tareas de clasificación. Es la relación entre las verdaderas (true positive, TP) y las extraídas por el modelo (elementos seleccionados).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

- **Exhaustividad (Recall)**, es la métrica de exhaustividad, es decir es la métrica que dice cuantas identidades verdaderas (true positive) el modelo logra identificar.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

- **F1-score**, puede resumirse como la media armónica entre la precisión y el recall.

$$F1_Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

- **Exactitud (Accuracy)** mide la proporción de aciertos globales del modelo. Se calcula mediante la proporción de predicciones correctas (positivas y negativas) sobre el total de predicciones realizadas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Estas métricas pueden ser aplicadas mediante, métricas macro o micro. Las Macro (macro-average) combinan los resultados de todas las clases antes de calcular la métrica, dando más peso a las clases mayoritarias, porque cada ejemplo cuenta igual. Y las Micro (micro-average) calculan la métrica por clase individual y luego hace un promedio simple, dando más importancia a las clases minorías, porque las trata igual que las grandes (Pedregosa et al., 2011; Kenneth Leung, 2022). En la Tabla 5 se pueden ver las ecuaciones para cada métricas aplicadas a Macro y Micro.

Tabla 5

Comparación entre métricas macro y micro.

Métrica	**Micro-average**	**Macro-average**
Precisión	$\frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)}$	$\frac{1}{N}\sum_i \frac{TP_i}{TP_i + FP_i}$
Recall	$\frac{\sum_i TP_i}{\sum_i (TP_i + FN_i)}$	$\frac{1}{N}\sum_i \frac{TP_i}{TP_i + FN_i}$
F1-score	$\frac{2 \cdot Precision_{micro} \cdot Recall_{micro}}{Precision_{micro} + Recall_{micro}}$	$\frac{1}{N}\sum_i \frac{2 \cdot Precision_i \cdot Recall_i}{Precision_i + Recall_i}$

Además de las métricas tradicionales de evaluación, pueden emplearse métricas de similitud entre conjuntos para comparar estructuras extraídas automáticamente con anotaciones de referencia como el Coeficiente de solapamiento (*Overlap coefficient*).

3.12.7 Validación

La validación de modelos *Deep Learning* se basa en métodos de re-muestreo (*resampling*), los cuales permiten estimar la capacidad predictiva en datos no vistos. Estos métodos consisten en dividir el conjunto de datos en subconjuntos de entrenamiento y validación, ajustar el modelo y evaluar su desempeño en las muestras restantes. Este proceso puede repetirse varias veces y los resultados se promedian para reducir la variabilidad asociada a la partición aleatoria de los datos. La principal diferencia entre los métodos de re-muestreo esta en la forma en que se generan estos subconjuntos (Rodrigo, 2011; James et al., 2013).

3.12.7.1 Hold-Out. El método *Hold-Out* consiste en dividir el conjunto de datos en particiones de entrenamiento y evaluación. Su principal limitación es que la calidad de la evaluación

depende fuertemente de la partición realizada, lo que puede generar resultados poco robustos cuando el *dataset* es pequeño.

3.12.7.2 Validación cruzada tipo K-Fold (k-Fold Cross Validation). En particular, cuando se tienen tamaños de datos no muy grandes se recomienda emplear *k-Fold Cross Validation* como método de validación de modelos. Funciona al dividir el *Dataset* en K grupos que se reutilizan para el entrenamiento y la validación. Y está en capacidad de disminuir la varianza en la estimación del error esperado, sin que el coste computacional sea muy elevado cuando se compara con otros métodos (Pedregosa et al., 2011). Cuenta con dos variantes, el convencional (k-Fold) y el estratificado (StratifiedKFold), como se ve en la Tabla 6 . También es considerado el método más robusto y que se puede utilizar para estimación y optimización de hiperparámetros de entrenamiento .

Tabla 6

Comparación entre KFold y StratifiedKFold.

Aspecto	KFold	StratifiedKFold
División de datos	*k* particiones de igual tamaño, sin considerar las clases	*k* particiones manteniendo la proporción de clases
Balance de clases	No garantizado	Garantizado, cada fold refleja la distribución original
Uso típico	Regresión o clasificación con clases balanceadas	Clasificación binaria o multiclase con clases desbalanceadas
Riesgo	Algunos folds pueden carecer de ejemplos de ciertas clases	Cada fold contiene representación adecuada de todas las clases
`scikit-learn`	`KFold(n_splits=k)`	`StratifiedKFold(n_splits=k)`

3.13 Recolección de datos de la Web (Web Data collection)

La recolección de datos corresponde al proceso de obtención y preparación de información proveniente de diferentes fuentes con el objetivo de ser utilizada en análisis, investigación o entrenamiento de modelos computacionales (Munzert et al., 2015). Entre las técnicas automatizadas más utilizadas para la adquisición de datos textuales cuando se trata de la web, es el raspado de web (*Web Scraping*).

En tareas de NLP, las técnicas de *Web Scraping* son ampliamente utilizadas para la construcción de conjuntos (*Datasets*) empleados en procesos de entrenamiento y evaluación de modelos (Jurafsky and Martin, 2026). El *Web Scraping* implica obtener y extraer datos de una página web. Las páginas web se construyen usando lenguajes de marcado basados en texto (HTML y XHTML), y frecuentemente contienen una gran cantidad de datos útiles en forma de texto. Sin embargo, la mayoría de las páginas web están diseñadas para usuarios humanos y no para facilitar el uso automatizado (El Asikri et al., 2020).

Dependiendo de los datos que se quieren recuperar y del comportamiento de la web de donde se van a recuperar los datos se puede decir que hay dos categorías de *Web Scraping*: A contenido estático, donde se recupera o solo hay contenido de HTML; es una recuperación mediante un análisis directo del HTML, usando librerías como *Beautiful Soup* (Richardson, 2026) o *Requests* (Reitz, 2026). Y a contenido dinámico, donde el contenido es generado mediante *JavaScript* y requiere herramientas capaces de simular la interacción de un navegador web como la librería *Selenium* (Project, 2026).

Además existen numerosas y variadas herramientas que nos permiten realizar el *Web Scraping*, estas se pueden dividir en tres grupos (Singrodia et al., 2019; El Asikri et al., 2020; Wahed et al., 2024):

1. *Bibliotecas para lenguajes de programación (Libraries)*: Uno de los enfoques más comunes, es el sistema de *Web Scraping* mediante la programación y creacion de un código, usando librerías que permiten acceder al sitio web mediante la implementación de protocolo HTTP, que luego usando funciones de software convencionales como, la comparación de expresiones regulares, tokenización y funciones propias del lenguaje de programación permiten extraer la información requerida. Este enfoque suele ser aplicado de forma individual para una web y requiere trabajo para implementarse y un mantenimiento continuo, pero ofrece una mayor automatización y control.

2. *Interfaces de programación de aplicaciones (API)*: Suelen ser aplicaciones de consola creadas por los mismos sitios Web que permiten hacer solicitudes HTTP para obtener datos en archivos estructurados como Excel o JSON. Es el método más recomendado cuando esta disponible, ya que requiere un menor trabajo de programación y es más estable que el anterior.

3. *Entornos basados en escritorio (Desktop-based)*: Estos suelen ser instalados en el sistema operativo y cuentan con interfase gráfica, o estar disponibles como complementos o extensiones de navegador. Generalmente permite al usuario dirigirse hacia la Web objetivo y seleccionar los elementos de la página a extraer. Estas soluciones no son totalmente automáticas y suelen tener una circulación comercial con acceso restringido.

Legalmente, el raspado web se sitúa en un panorama complejo con diversas leyes y regulaciones internacionales, lo que a menudo conduce a ambigüedades en el cumplimiento y la aplicación. Por esto el uso de técnicas de web scraping debe realizarse considerando aspectos éticos y legales relacionados con derechos de autor, políticas de uso de los sitios web y protección de datos. Asimismo, es recomendable respetar mecanismos de restricción como archivos *robots.txt* y evitar la recolección de información sensible o privada (Essien et al., 2023).

4 Estado del Arte

4.1 NLP convencional

El NLP nació mediante el uso de reglas heurísticas particulares para cada aplicación. Luego apareció el enfoque tradicional donde los textos son preprocesados y representados mediante vectores de características que pasan a algoritmos de Aprendizaje de Maquina (*Machine Learning*, ML) encargados de aprender un mapeo entre los datos de entrada y las salidas (Wankmüller, 2022). Actualmente este enfoque se considera en des-uso para aplicaciones complejas de la vida real y muestra desventajas como, el uso de representaciones basadas en vectores de características, que al requerir la definición previa de los atributos relevantes, puede complicar el diseño del modelo. En tareas de procesamiento de texto, estas representaciones suelen generar alta dimensionalidad en escenarios con pocos datos etiquetados, lo que puede conducir a la maldición de la dimensionalidad (Bengio et al., 2003). Asimismo, al no considerar el orden de las palabras, ni las relaciones sintácticas o semánticas, se limita la capacidad del modelo para capturar adecuadamente el significado del texto (Turney and Pantel, 2010; Wankmüller, 2022).

4.2 NLP con Deep learning

Actualmente gracias a las características de lo problemas a resolver en el análisis de texto, el NLP es aplicado mediante técnicas basadas en Aprendizaje profundo (*Deep Learning*, DL), que tienen mayor capacidad para modelar la complejidad del lenguaje (Alias and Cassanelli, 2019). Aun así, estos modelos ofrecen algunas desventajas como, el conducir a modelos más complejos

de lo necesario cuando se usan a conjuntos de datos pequeños. Que el desempeño dependa fuertemente de la disponibilidad de grandes volúmenes de datos de entrenamiento, incrementando así, los costos computacionales, el consumo de tiempo y la dificultad de obtener datos adecuadamente anotados (Zhao et al., 2024). También, su adaptación en nuevas tareas o dominios suele requerir la recolección y preparación de nuevos conjuntos de datos, llegando a dificultar la transferencia entre distintas disciplinas cuando hay grandes variaciones en las características, el contenido, la gramática y/o el vocabulario (Wankmüller, 2022).

4.3 Modelos PLMS basados en arquitectura Tranformers

Uno de los artículos más influyentes en el campo de la NLP fue la primera introducción de los *Transformers* (Vaswani et al., 2017), un tipo más moderno de modelo *Deep Learning*. Actualmente existen modelos de lenguaje a gran escala (LLM) como BERT (Chakkarwar et al., 2023), un modelo de inteligencia artificial desarrollado por *Google*, que se especializan en la generación de texto, resúmenes y chat-bots. SciBERT (Beltagy et al., 2019) un modelo de lenguaje basado en BERT, entrenado específicamente en textos científicos para mejorar el rendimiento en tareas de NLP en dominios académicos. Este tipo de modelos tienen aplicaciones más amplias, pero requieren importantes recursos informáticos para su pre-entrenamiento o adaptación para algunas tareasHong (2023).

4.4 NLP para la extracción de información y análisis de artículos científicos

Han sido empleados para la extracción automática de información en diversos dominios, como la tecnología (como en Li et al. (2019b)), medicina (como en Hu et al. (2024a); Yu et al.

(2024)), la ciencia de materiales (como en Beard and Cole (2022); Hong (2023)) y la química (como en Krallinger et al. (2017)).

Se han empleado modelos convencionales de *aprendizaje profundo* (DL) para extraer información de las celdas solares de perovskita, como se ve en Beard and Cole (2022), que aplica el kit de herramientas de minería de texto de *ChemDataExtractor* Swain and Cole (2016). En Makino et al. (2022) se centran en la extracción de diferentes materiales utilizando los primeros modelos de NLP propuestos, los Modelos basados en reglas. Además, existen varias herramientas centradas en diferentes aplicaciones y que emplean diferentes tipos de modelos, como:

- Herramientas de ML convencional: *OSCAR4* (Jessop et al., 2011), con un diseño adecuado para la extracción de información química variada, y *ChemicalTagger* (Hawizy et al., 2011), diseñado para anotar y etiquetar procedimientos experimentales en el ámbito de la química.

- Herramientas de *Deep Learning*: *spaCy* (Honnibal et al., 2026a), framework de procesamiento de lenguaje natural orientado a aplicaciones en producción, que integra modelos estadísticos y redes neuronales dentro de *Pipelines* de procesamiento, en con la capacidad de incorporar diferentes enfoques DL y modelos preentrenados de lenguaje (PLMs). Y *Stanza* (Qi et al., 2020), biblioteca para NLP que utiliza redes neuronales profundas.

- Herramientas de PLMs: *AllenNLP* (Gardner et al., 2018) que proporciona una plataforma modular para el entrenamiento y la evaluación de modelos de NLP avanzados, y *Hugging Face Transformers* (Wolf et al., 2020b), biblioteca con diferentes arquitecturas Transformer de última generación diseñadas bajo una API unificada.

Todos estas herramientas arrojan buenos resultados, pero a menudo requieren un extenso trabajo manual de adaptación o modificación, pre-entrenamiento y entrenamiento. Además, se han encontrado los artículos Dagdelen et al. (2024) y Danenas and Skersys (2022) que exploran y comparan diferentes modelos NLP. Se encontró que el modelo tradicional *Stanza* (Qi et al., 2020) muestra mejores resultados en tareas simples de NER en comparación con los LLMs, y que a pesar de eso los modelos LLMs muestran ser suficientes para esa tarea, y tener mejores resultados para la determinación o clasificación de relaciones entre entidades en los casos "*donde la información no puede modelarse como simples relaciones por pares entre entidades. Esto es particularmente evidente en la ciencia de materiales inorgánicos, donde las propiedades de un compuesto están determinadas por una combinación compleja de su composición elemental, geometría atómica, micro-estructura, morfología*" (Dagdelen et al., 2024).

En el campo de la ciencia de materiales se ha encontrado el uso de modelos LLMs en diversas áreas como, *MatSciBERT* (Gupta et al., 2022) que esta basado en *BERT* (Chakkarwar et al., 2023) disponible a traves de la biblioteca Hugging Face Transformer (Wolf et al., 2020b), que tiene la capacidad para comprender y generar texto relacionado con ciencia de materiales, la extracción de entidades (NER), la clasificación de artículos científicos y la predicción de propiedades de materiales. Además de una gran facilidad de generalizar nuevos conjuntos de datos y adaptarse a diferentes tareas dentro del campo de la ciencia de materiales. Mostrando que estos modelos ofrecen la ventaja de requerir menos trabajo de desarrollo y demostrando una gran sensibilidad al contexto a la hora de extraer información y resultados de vanguardia en las tareas de reconocimiento de entidades (NER), clasificación de relaciones (RE) y clasificación abstracta.

Por otro lado, con el modelo *BatteryDataExtractor* (Huang and Cole, 2022) desarrollado para la extracción de material de baterías, se observa que se pueden mejorar los resultados desarrollando y/o implementando aplicaciones con modelos combinados, que aprovechen lo mejor de los diferentes modelos de NLP para áreas o temas específicos, teniendo en cuenta las limitaciones que surgen de la necesidad de GPU de alto rendimiento en algunas aplicaciones.

4.5 Transfer learning en modelos PLMs para la extracción de información

Existen varios artículos que usan diferentes métodos de *Transfer learning* para la extracción de información desde datos estructurados, semi-estructurados y no estructurados. En Lee et al. (2017) usaron *Transfer learning* en la tarea de Des-identificación de pacientes. En McCann et al. (2017) probaron la transferibilidad de vectores de palabras entrenados en conjuntos de datos de traducción automática con diferentes tamaños. En Lin et al. (2020) realizaron experimentos para estudiar la viabilidad de aplicar *Transfer learning* para desarrollar rápidamente un sistema para procesar informes hospitalarios de diferentes fuentes. En Nguyen et al. (2020) utilizan técnicas de *Transfer learning* para mejorar la extracción de información cuando se dispone de datos limitados, usando BERT como modelo preentrenado. Y en Wankmüller (2022) se comparó el uso del tranfer learning, entre tres modelos basados en transformadores (BERT, RoBERTa y Longformer), contra algoritmos de aprendizaje automático convencionales, en tres conjuntos de datos relacionados con las ciencias sociales. Todos estos artículos demuestran las ventajas de esta técnica respecto a crear o entrenar modelos desde cero, especialmente cuando no se cuenta con una gran cantidad de artículos o datos para alimentar al modelo.

4.6 NLP para la extracción y análisis de información aplicado a celdas solares de perovskita

Jiménez-Díaz et al. (2022), una tesis de pregrado, y Jiménez-Díaz et al. (2025), un artículo publicado posteriormente durante el desarrollo de este trabajo, emplean similar metodología de extracción de entidades nombradas (NER) en PSCs, utilizando un modelo de *Deep Learning* basado en la librería *SpaCy*; donde el artículo introduce modificaciones y mejoras en los algoritmos, y valida la propuesta sobre una base de datos distinta. Beard and Cole (2022) detalla el uso de *ChemDataExtractor* como modelo base para la generación automática de bases de datos de PSCs, mediante la extracción de entidades nombradas (NER), demostrando cómo la automatización puede acelerar la recopilación y creación de bases de datos científicas.

En Xie et al. (2024) se centran en la creación de un conjunto de datos estructurado para PSCs y la predicción del rendimiento, utilizando modelos LLMs y redes convolucionales. Shabih et al. (2025) usa modelos LLMs para extraer información de PSCs en artículos de PDF mostrando métricas prometedoras, pero la validación sobre un corpus reducido (15 artículos) limita la generalización de los resultados. Por último, en Liu et al. (2025) hace uso de grafos de conocimiento y modelos LLMs para construir una infraestructura completa: grafo, *Datasets* y modelos especializados, para extraer información de PSCs. Estos trabajos demuestran el potencial de los LLMs para la extracción automática de información científica y muestran cómo, la combinación de datos estructurados, modelos NLP y mecanismos de representación del conocimiento puede facilitar la extracción, organización y aprovechamiento de información científica en el campo de las celdas solares. Sin embargo, el uso frecuente de modelos de gran escala (LLMs) puede dificultar su

adopción en entornos con recursos computacionales limitados, lo que motiva la exploración de alternativas basadas en PLMs más compactos y técnicas de ajuste fino.

Así, Göpfert et al. (2026) presenta un marco generalista para la extracción de información cuantitativa desde literatura científica, empleando modelos de lenguaje preentrenados (PLMs) de acceso abierto y menor tamaño. El sistema realiza una extracción contextualizada, identificando no solo valores numéricos y unidades, sino también las propiedades asociadas y, en algunos casos, los calificadores que describen las condiciones de medición (tiempo, espacio, método, referencia). A pesar de su menor tamaño, estos modelos alcanzan métricas de evaluación comparables a las de LLMs más grandes, lo que demuestra su eficiencia y potencial para tareas extracción científica. Aunque las Perovskitas no constituyen el foco principal del estudio, se incluyen como caso de aplicación en la extracción de *Band gaps*, lo que evidencia la aplicabilidad transversal de este enfoque en el ámbito de las celdas solares de Perovskita.

Además, trabajos como Vepreva et al. (2026) enfatizan la necesidad de *Datasets* manuales y validados por expertos para evaluar la extracción de información química. En este caso, los autores recopilaron y curaron datos multimodales (texto, tablas y gráficos) provenientes de artículos científicos sobre nanomateriales y moléculas pequeñas, asegurando su consistencia mediante validación humana. Estos conjuntos permiten probar distintos modelos de extracción bajo las mismas condiciones, facilitando comparaciones justas y reproducibles entre enfoques tradicionales y modelos de lenguaje. Aunque no se aplican directamente a Perovskitas, aportan lineamientos metodológicos valiosos sobre cómo garantizar la calidad, transparencia y reproducibilidad de datos científicos.

Disponiendo toda esta información, este trabajo propone un *Pipeline* híbrido para la extracción de información en celdas solares de Perovskita, basado en aprendizaje por transferencia (*Transfer Learning*) mediante el uso de modelos de aprendizaje profundo (*Deep Learning*), como *spaCY* y el PLM, BERT, complementados con técnicas de Ingeniería del Conocimiento (*Knowledge Engineering*) para las etapas de post-procesamiento. A diferencia de enfoques recientes fundamentados en LLMs de gran escala, la propuesta busca ofrecer una alternativa viable para entornos con recursos computacionales limitados, manteniendo la capacidad de extraer información especializada desde literatura científica.

5 Diseño del Pipeline de extracción y estructuración de información aplicado a celdas solares de Perovskita

Figura 11

Diagrama resumen del Pipeline (Tubería) desarrollado.



En este trabajo se aplica un *Pipeline* que usa un enfoque híbrido mediante la aplicación del *Transfer Learning* en dos fases, usando modelos *Deep Learning*, con *spaCy* para la tarea NER y con el PLM BERT para la tarea RE, complementados con técnicas de Ingeniería del Conocimiento (*Knowledge Engineering*) para la etapa de post-procesamiento. El proceso se centra en la extracción contextualizada de información relevante sobre celdas solares de Perovskita, mediante la agrupación de entidades asociadas a un mismo dispositivo (*device*). Para cada dispositivo se recopila la información estructural de las capas ETL, HTL y Perovskita, los parámetros de desempeño tales como V_{oc}, J_{sc}, PCE y FF, e información relacionada con los procesos de fabricación de

la celda. La Figura 11 presenta la estructura general del Pipeline propuesto y las Figuras 12, 13, 14, 15, 19, 21, 24 y 27 ilustran la representación conceptual de cada proceso.

Aunque este trabajo toma como referencia los códigos del módulo de extracción de entidades (NER) desarrollado en Jiménez-Díaz et al. (2022), estos fueron adaptados, actualizados y extendidos para satisfacer los nuevos requerimientos de la investigación. En particular, este trabajo incorpora etapas para la extracción de entidades (NER), la extracción de relaciones (RE), la construcción y anotación del corpus, el entrenamiento y evaluación de modelos PLM usando aprendizaje por transferencia, así como mecanismos de post-procesamiento orientados a la agrupación de información por dispositivos de celdas solares de Perovskita. De esta manera, el trabajo previo sirvió como base solo para el componente NER, mientras que la presente propuesta amplía significativamente el alcance metodológico y funcional del sistema, tal como se resume en la Figura 11.

Se limita la selección de entidades nombradas a 16 etiquetas como se ve en la Tabla 7 y se limitan a 5 clases, las relaciones posibles entre las entidades nombradas como se ve en la Tabla 8.

Tabla 7

Entidades nombradas definidas para la extracción de información en celdas solares de Perovskita.

Categoría	**Etiqueta**	**Descripción**
Capas de la celda	ETL	Capa de transporte de electrones (*Electron Transport Layer*)
	HTL	Capa de transporte de huecos (*Hole Transport Layer*)
	PTF	Capa de Perovskita

Categoría	Etiqueta	Descripción
Parámetros de desempeño	VOC	Voltaje de circuito abierto
	PCE	Eficiencia de conversión de potencia
	JSC	Corriente de cortocircuito
	FF	Factor de forma
Valores de los parámetros de desempeño	NVOC	Valor asociado al parámetro VOC
	NPCE	Valor asociado al parámetro PCE
	NJSC	Valor asociado al parámetro JSC
	NFF	Valor asociado al parámetro FF
Métodos de fabricación	MTDS	Nombre del método de fabricación
	ENMTDS	Condiciones ambientales o de entorno asociadas al método
	VMTDS	Variables asociadas al método de fabricación (temperatura, presión, etc.)
	NMTDS	Valores de las variables asociadas al método de fabricación

Tabla 8

Relaciones definidas para la extracción de información en celdas solares de Perovskita.

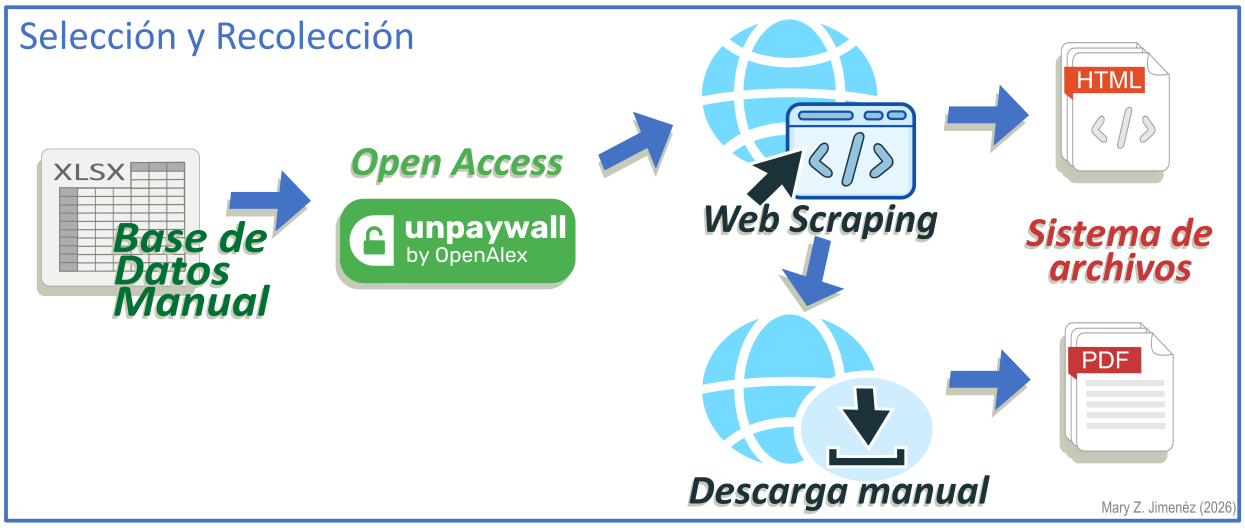
Categoría	Etiqueta	Pares de entidades
Relaciones estructurales o de uso	USED_WITH	(HTL, PTF), (ETL, PTF), (ETL, HTL)
Relaciones de desempeño	HAS_VALUE	(PTF, VOC), (HTL, VOC), (ETL, VOC), (PTF, PCE), (HTL, PCE), (ETL, PCE), (PTF, JSC), (HTL, JSC), (ETL, JSC), (PTF, FF), (HTL, FF), (ETL, FF)
Relaciones de proceso o fabricación	PREPARED_WITH	(HTL, MTDS), (ETL, MTDS), (PTF, MTDS)

Categoría	Etiqueta	Pares de entidades
Condiciones de proceso	PERFORMED_ UNDER_ CONDITION	(MTDS, ENMTDS), (MTDS, VMTDS), (MTDS, NMTDS), (VMTDS, NMTDS)
Relación negativa o sin relación	NO_RELATION	Entidades sin relación semántica definida.

5.1 Selección y Recolección de los datos

Figura 12

Representación conceptual del proceso selección y Recolección de los datos.



5.1.1 Selección del Dataset

Para crear el conjunto de datos (*Dataset*) de entrenamiento, evaluación y validación de este trabajo, la recolección de documentos fue realizada durante el año 2024 únicamente sobre recursos de acceso abierto (*Open Access*, OA).

Para la identificación de artículos de acceso abierto (OA), se descarga un subconjunto de la base de datos del proyecto *The Perovskite Database Project*[3] (Jacobsson et al., 2022). El subconjunto utilizado contiene 29070 registros correspondientes a 5107 artículos sobre celdas solares de perovskita convencionales, publicados entre los años 2003 y 2020. Posteriormente, la información fue almacenada en un archivo Excel y se extrajo la columna correspondiente a los identificadores DOI de los artículos.

A continuación, se emplea la API REST de la base de datos *Unpywall*, ampliamente utilizada para consultar información sobre acceso abierto a partir de identificadores DOI. Se genera un archivo EXCEL que contiene información sobre el estado de acceso abierto, el tipo de licencia, la URL de acceso y otros metadatos relevantes. Mediante este procedimiento se identificaron 1308 artículos OA (1313 inicialmente encontrados, de los cuales 5 presentaron problemas durante el procesamiento). Debido a cambios en las políticas editoriales y en la disponibilidad de repositorios, el estado de acceso abierto de algunos documentos puede variar con el tiempo, por eso se genera un archivo que registra la información de los artículos procesados.

5.1.2 Recolección de los datos mediante Web Scraping

La recolección de datos de los 1308 artículos OA encontrados se realizo entre finales de 2024 y principios de 2025 de forma semi-automática mediante *Web Scraping* con código propio y diferentes librerías de Python. Para descargar los archivos en formato HTML/XML y guardarlos en una carpeta se desarrolló un Script que aplica técnicas de extracción de contenido estático,

---

[3] Los datos pueden descargarse desde `https://www.perovskitedatabase.com/`

con las librerías *Beautiful Soup* (Richardson, 2026) y *Requests* (Reitz, 2026). Adicionalmente, el script genera un archivo de registro con información resumida del proceso de descarga, incluyendo artículos procesados, descargas exitosas y archivos que no se pudieron descargar.

El script fue diseñado para realizar la descarga de los artículos en bloques, incorporando tiempos de espera (re-conexión) entre solicitudes, con el fin de reducir errores de conexión, bloqueos y permitir la supervisión Humana. Para resolver los inconvenientes causados por los *Captchas* el código le sede temporalmente el control a usuario y luego continua con la descarga del siguiente artículo. Pero, debido a la naturaleza dinámica de las plataformas web y cambios posteriores en políticas de acceso o mecanismos anti-automatización, algunos recursos pueden no conservar actualmente las mismas condiciones de acceso.

Durante el desarrollo del trabajo se encontró que los artículos de acceso abierto (OA, *Open Access*) estaban disponibles en dos formatos: HTML y PDF. No se dispuso de documentos en formato XML como se había previsto inicialmente en los objetivos. El procesamiento se adaptó a los formatos efectivamente disponibles.

Para determinar cuáles documentos HTML contenían el texto completo del artículo, se implementó un código que verificó los 1308 archivos HTML descargados. Mediante la librería *Beautiful Soup*, se establecieron condiciones basadas en elementos HTML característicos de cada revista o editorial (*journal*) para identificar si el contenido disponible incluía secciones adicionales al resumen (*Abstract*). A partir de la lista de archivos OA (*ARTopen_Perovskite.xlsx*) se identifican los HTMLs que no contienen el documento completo, creando una lista de documentos en formato PDF que se deben descargar manualmente de la web. Esta lista, junto con un resumen del proceso

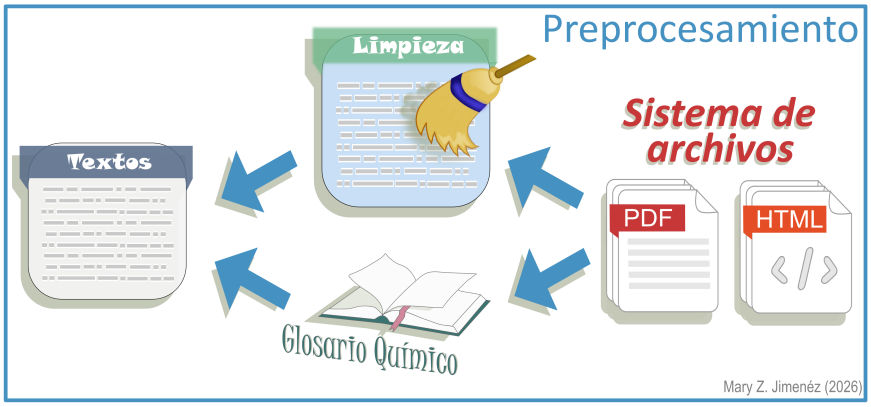
de verificación y del número de archivos pendientes a descarga, se almacena en el archivo de Excel (*ARTopen_Perovskite_pdfs.xlsx*). Como resultado, se identifican 681 artículos cuyo documento completo se encontraba disponible únicamente en formato PDF, los cuales fueron descargados manualmente.

Por último, se verifican y descargan manualmente los 5 artículos que no pudieron ser recuperados y la lista de 681 artículos identificados como formato PDF. Obteniendo 1309 documentos en formatos PDF y HTML, los cuales fueron organizados y almacenados mediante una estructura de directorios en la carpeta *DATA_TRAIN*.

5.2 Preprocesamiento de los datos

Figura 13

Representación conceptual del preprocesamiento de los datos.



Los artículos científicos sobre celdas solares de perovskita contienen una gran cantidad de terminología especializada, abreviaturas, fórmulas químicas y denominaciones particulares que pueden variar entre publicaciones. Adicionalmente, es común encontrar compuestos o materiales

identificados mediante nombres específicos definidos por los propios autores, lo que dificulta la comparación automática de información entre documentos.

Por otra parte, durante la extracción de texto desde archivos PDF y HTML pueden presentarse inconsistencias asociadas a la codificación de caracteres ASCII extendida (IBM, 2014), símbolos especiales y formatos de representación utilizados por diferentes editoriales científicas. Estas variaciones pueden introducir errores durante el almacenamiento y procesamiento de la información en formato de texto plano.

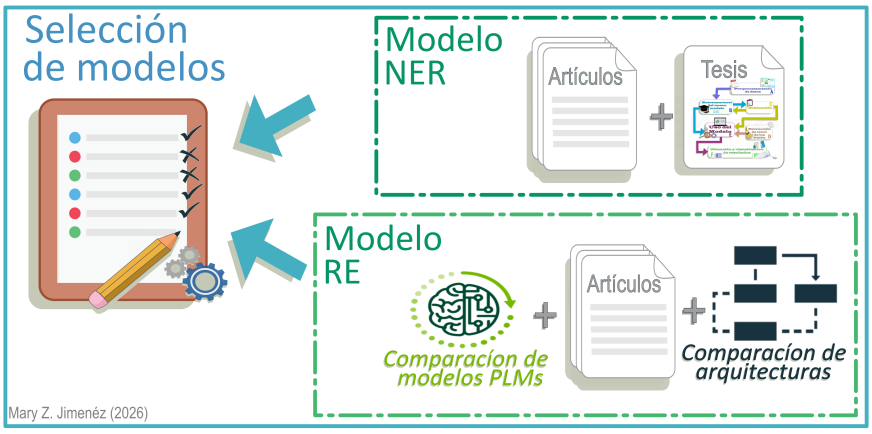
Con el fin de utilizar los documentos en las etapas de entrenamiento, evaluación y validación de los modelos, así como en las diferentes fases del *Pipeline*, se implementó un proceso de normalización textual. Permite convertir el contenido extraído a un formato homogéneo, estandarizando expresiones, unidades, abreviaturas, símbolos y otras variaciones frecuentes presentes en los artículos científicos. Se emplea un archivo de configuración en formato JSON (*Replaces.json*), el cual contiene reglas de sustitución para símbolos, términos y expresiones identificadas durante el análisis de los datos.

5.3 Selección de modelos

Con el sistema de archivos con documentos en PDF y HTML, y el pre-procesamiento de los datos establecido, se seleccionan los modelos *Deep Learning* preentrenados que se van a usar para el desarrollo de la tarea de Reconocimiento de Entidades Nombradas (NER) y de Extracción de Relaciones (RE). La selección de los modelos *Deep Learning* se realiza previo a la construcción del corpus etiquetado porque que la estructura y formato de los datos de entrenamiento, dependen de los requerimientos específicos de cada arquitectura.

Figura 14

Representación conceptual del proceso de selección de los modelos NER y RE.



5.3.1 Selección del modelo NER

La herramienta *SpaCy* (Honnibal et al., 2015) fue seleccionada para la tarea de NER considerando la experiencia obtenida en el proyecto previo (Jiménez-Díaz et al., 2022), así como la evidencia reportada en la literatura científica de artículos como, Neumann et al. (2019), Shelar et al. (2020), Naseer et al. (2021) y Lee et al. (2023), que muestran los buenos resultados, velocidad, robustez y rendimiento que se puede obtener en la tarea NER y otras de sus sub-tareas, cuando se usa la librería *SpaCy* respecto a otras librerías.

SpaCy es una biblioteca de código abierto orientada al desarrollo de tareas de procesamiento de lenguaje natural, incluyendo reconocimiento de entidades nombradas, análisis sintáctico, lematizacion, clasificación de texto y segmentación de oraciones. En su versión 3.6 cuenta con una arquitectura modular que permite integrar modelos estadísticos clásicos del *Machine Learning* y modelos basados en *Deep Learning*, así como realizar procesos de entrenamiento y ajuste sobre modelos preentrenados adaptados a tareas específicas (Honnibal et al., 2026a).

5.3.2 Selección del modelo RE

Buscando minimizar el uso de recursos computacionales se decide utilizar el método más sencillo y flexible para aplicación de la tarea RE. Se evalúan diferentes PLMs preentrenados con distinto número de parámetros, buscando identificar el modelo que ofrece un equilibrio entre desempeño y consumo de recursos. Los modelos candidatos y demás detalles de la comparación se describen en la Sección 6.4 de resultados.

Además, trabajos como Alyafeai et al. (2020) y Tinn et al. (2023) comparan el desempeño de modelos PLMs y LLMs en diversas tareas de NLP, concluyendo que los modelos basados en BERT continúan siendo una alternativa competitiva para tareas de comprensión y extracción de información, especialmente cuando no se requiere generación de texto. En contraste, los LLMs suelen destacar en tareas generativas debido a sus capacidades de producción de lenguaje natural.

El estudio "*A Survey on Cutting-Edge Relation Extraction Techniques Based on Language Models*" (Diaz-Garcia and Lopez, 2025) analiza las principales técnicas y modelos utilizados en la tarea de extracción de relaciones (RE). Los autores identifican una fuerte presencia de enfoques basados en BERT dentro de la literatura reciente, atribuyéndola a su eficacia para alcanzar resultados competitivos y a su amplia adopción en sistemas de extracción de relaciones.

Tras la evaluación comparativa y la lectura de artículos, se selecciona para la tarea de RE el modelo SciBERT (Beltagy et al., 2019), un modelo PLM preentrenado basado en BERT, adaptado a textos científicos mediante 1.14 millones de artículos de *Semantic Scholar*. Al igual que BERT-base cuenta con 12 capas de bloques *Transformers* que tienen atención bidireccional, es decir, el

modelo no solo mira las palabras anteriores (como en modelos tipo RNN o GPT), sino también las posteriores, de manera que cada token se representa considerando todo el contexto de la oración. Existen diferentes versiones como se ve en la Tabla 9.

Tabla 9

Principales versiones de modelo preentrenado SciBERT.

Modelo	Vocabulario	Sensibilidad a mayúsculas	Ventaja
scivocab_uncased	Científico (Scivocab)	No	Mejor adaptación a terminología científica con mayor robustez frente a variaciones ortográficas.
scivocab_cased	Científico (Scivocab)	Sí	Conserva información útil en fórmulas químicas, siglas y nomenclatura técnica.
basevocab_uncased	General (BERT original)	No	Sirve como referencia frente a vocabulario no especializado.
basevocab_cased	General (BERT original)	Sí	Permite comparar el impacto de preservar capitalización sin vocabulario científico específico.

5.4 Generación del conjunto de datos etiquetados

Para poder aplicar el método de *Fine Tuning* del *Transfer Learning* mediante un aprendizaje supervisado a modelos *Deep Learning*, se debe crear un *Dataset* etiquetado para el entrenamiento, evaluación y validación de los modelos, y procesar el texto correctamente para poder aplicar el modelo.

Figura 15

Representación conceptual del proceso de generación del conjunto de datos etiquetados para los modelos NER y RE.



5.4.1 Generación de la primer versión del Dataset para la tarea NER

Teniendo el archivo de estandarización del texto (*Replaces.json*) se desarrolla un modulo para la extracción, limpieza y segmentación del texto, con el fin de generar el conjunto de datos etiquetados (oraciones junto a sus entidades nombradas) en un JSON. La estructura necesaria para el entrenamiento de un modelo de la librería *Spacy* adaptado a la tarea de reconocimiento de entidades nombradas (NER), es una lista de listas, como se ve en el Apéndice E.1 y en el ejemplo de la Figura 16.

Figura 16

Ejemplo de una oración con sus entidades nombradas del dataset de entrenamiento NER.

```
["The solar cell achieved a PCE of 17.70%  with a Voc of 1.10 V, a Jsc of
 21.21 mA cm-2, and an FF of 75.55%  when measured under reverse voltage
 scanning.",
    {"entities": [[48,51,"VOC"],[65,68,"JSC"],[26,29,"PCE"],[94,96,"FF"],
                  [55,61,"NVOC"],[72,80,"NJSC"],[33,39,"NPCE"],
                  [100,106,"NFF"]]
    }
],
```

Nota. Se muestra la estructura de los datos en formato JSON.

Para obtener los datos etiquetados, se integraron diferentes módulos y funciones basados en las bibliotecas PyMuPDF (Liu and McKie, 2021), *Beautiful Soup* (Richardson, 2026) y *spaCy* (Honnibal et al., 2015). El sistema extrae automáticamente el texto de los artículos científicos, lo segmenta en oraciones y utiliza listas de términos y patrones previamente definidos para identificar y etiquetar las entidades nombradas de interés. La biblioteca *spaCy* permite incorporar nuevas categorías de entidades sin necesidad de entrenar previamente un modelo NER, facilitando así la construcción inicial del *Dataset*. El Apéndice A.1 presenta una descripción más detallada del proceso.

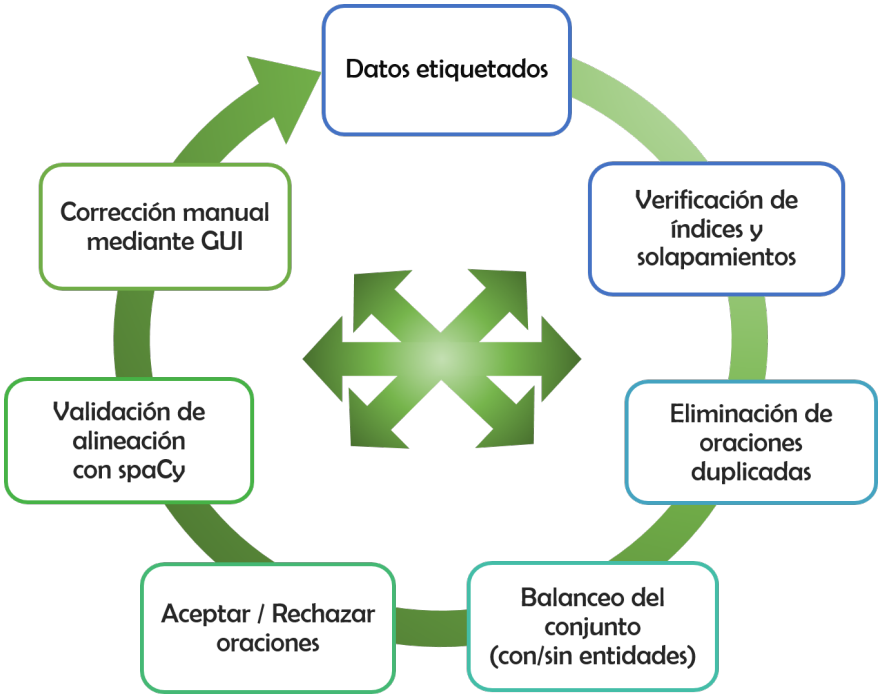
5.4.2 Depuración del Dataset para la tarea NER

En el proceso automático realizado en la sub-sección anterior las oraciones seleccionadas y el etiquetado de entidades con *spaCy* no es tan confiable, ya que permite la aparición de entidades nombradas repetidas, el mal etiquetado de algunas entidades, deficiencias en el posicionamiento de entidades, errores en tokenizador, oraciones demasiado largas, entre otras. Demostrando la ne-

cesidad de aplicar *Fine Tuning* para afinar un modelo preentrenado en aplicaciones especificas y de aplicar un proceso semi-automático de depuración del *Dataset*, donde el usuario tenga el poder de decisión y corrección.

Figura 17

Representación grafica del proceso de corrección y mejora de los datos etiquetados del Dataset.



Para la Depuración del *Dataset* se desarrolla un modulo que mediante diferentes funciones y decisiones del usuario, se corrigen y mejoran los datos. Las diferentes funciones pueden ser usadas o no, tantas veces como se considere necesario para obtener una versión final *Dataset* aceptable. La Figura 17 ilustra el proceso y el Apéndice A.2 presenta una descripción más detallada del proceso.

En este trabajo, al iterar sobre las diferentes funciones, se pasa de 149181 a 4082 oraciones, obteniendo el *Dataset* para el entrenamiento, evaluación y validación del modelo NER. Distribuido: 2927 (71.7 %) oraciones con una o más entidades y 1155 (28.3 %) oraciones sin entidades.

5.4.3 Conversión y depuración del Dataset para la tarea RE

Para aplicar la estrategia de entrenamiento de *Fine Tuning* en la tarea RE usando el modelo preentrenado SciBERT, es necesario primero convertir los datos etiquetados de la tarea NER, al formato necesario para la tarea de clasificación. Este formato esta compuesto de una lista de diccionarios de oraciones junto a sus pares entidades-relación, como se ve en el Apéndice E.2. En la Figura 18 se puede ver un ejemplo.

Figura 18

Ejemplo de una oración con su pareja entidades-relacion del Dataset de entrenamiento RE.

```
{"text": "The PCE of the PSCs with a simple structure (FTO  /  compact
<e1>TiO2</e1> / MAPbI3 / carbon film, where FTO is fluorine-doped
tin oxide) was improved from <e2>11%</e2> (without CuPc) to 15%
(with CuPc).",
    "label": "HAS_VALUE"
},
```

Nota. Se muestra la estructura de los datos en formato JSON.

Mediante un sistema semiautomático se construye el conjunto de datos etiquetados utilizado para el entrenamiento, evacuación y validación del modelo extracción de relaciones (RE). A partir de las entidades identificadas en el *Dataset* de NER, el sistema genera automáticamente una propuesta inicial de relaciones para cada oración. Y posteriormente, el usuario revisa y corrige estas relaciones con el apoyo de diferentes funciones de validación, obteniendo así el conjunto de datos final empleado del modelo RE. El Apéndice A.3 presenta una descripción más detallada del proceso.

Con este proceso se convierte el *Dataset* al formato correcto para usar el modelo preentrenado SciBERT. El *Dataset* contiene 5900 pares entidades-relación, donde 4019 (68.12 %) son NO_RELATION y 1881 (31.88 %) tiene alguna clase de relación.

5.4.4 Generación del Dataset para la evaluación (Test) independiente

Se aplica nuevamente el procedimiento empleado para los datos etiquetados de entrenamiento (Sub-Secciones 5.4.1,5.4.2 y 5.4.3) para generar un conjunto de datos etiquetados independientes con el propósito de medir el desempeño de los modelos NER y RE, frente a datos externos que no participaron en las etapas de desarrollo: selección del modelo, entrenamiento, ajuste de Hiperparómetros, ni validación.

Este proceso se realiza sobre 582 artículos en formato PDF de la base de datos (*Database*) proporcionada por el articulo Odabaşı and Yıldırım (2019), una colección diferente a la que se utiliza durante el entrenamiento, exclusivamente para la evaluación final de los modelos.

5.5 Entrenamiento de modelos Deep Learning para Information Extraction

En este trabajo el entrenamiento de los modelos NER y RE se realiza aplicando aprendizaje supervisado, usando modelos preentrenados y mediante un *Proceso de selección, optimización y validación del modelo* que sigue una serie de pasos generales, como se ilustra en la Figura 20 y se describe con mayor detalle en el Apéndice C.

El conjunto de datos utilizado para el desarrollo de los modelos usando el proceso de la Figura 20, se divide en un porcentaje para entrenamiento y otro porcentaje para validación. El conjunto de validación se utiliza para la evaluación de validación de los modelos durante el proceso de la Figura 20. Y el conjunto de datos completo se usa en la validación cruzada K-fold.

Figura 19

Representación conceptual del proceso de entrenamiento de los modelos NER y RE.

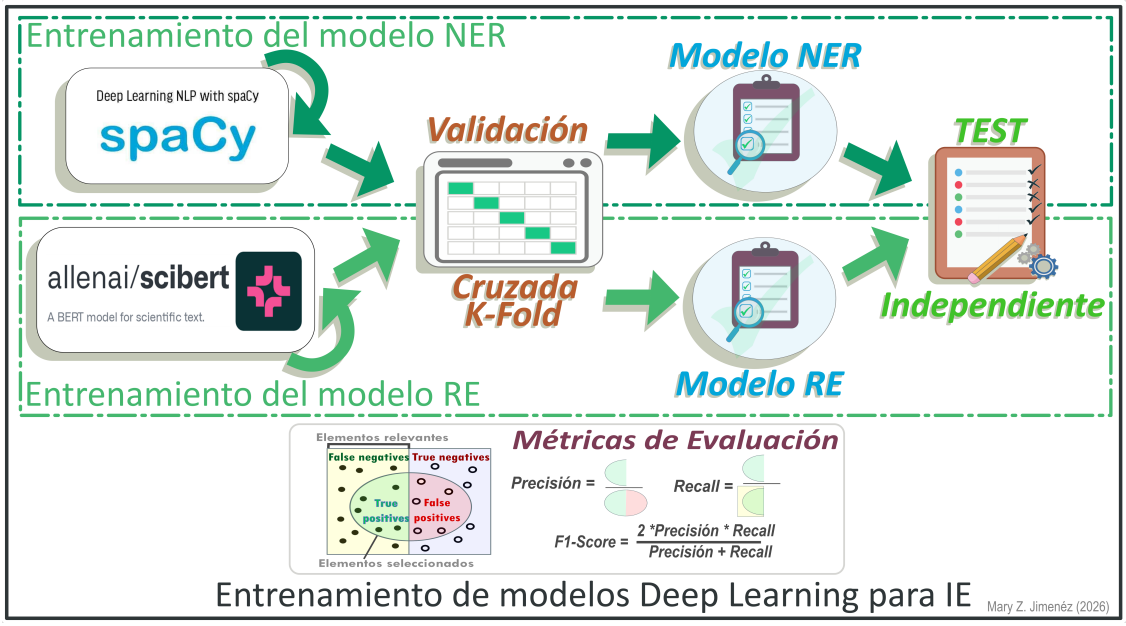
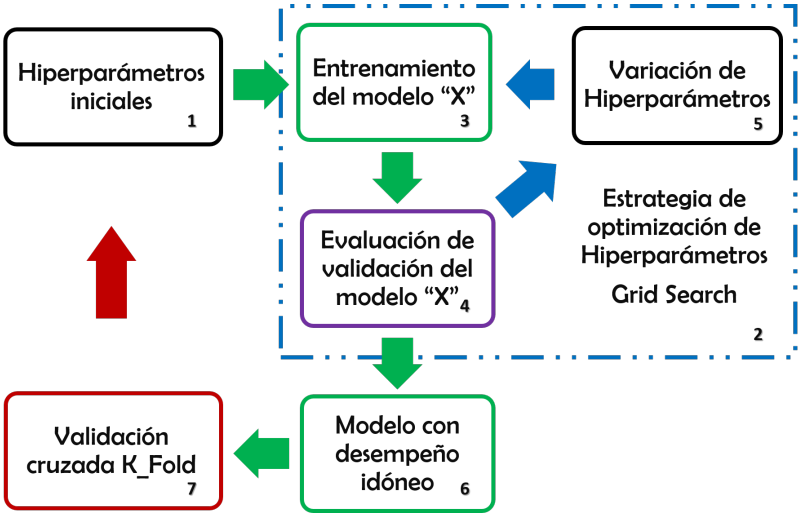


Figura 20

Esquema general del Proceso de selección, optimización y validación del modelo.



Nota. Flujo de trabajo utilizado en este trabajo para el entrenamiento supervisado de modelos de aprendizaje profundo (DL) en Python.

5.5.1 Evaluación mediante un conjunto de prueba (Test) independiente

Una vez finalizado el *Proceso de selección, optimización y validación del modelo* como se ve en la Figura 20 y seleccionado el modelo final, se realiza una evaluación adicional de los modelos NER y RE, sobre el conjunto de datos etiquetados (*Dataset*) de prueba independiente descrito en la Sección 5.4.4. Este conjunto se utiliza solo para la evaluación final de los modelos y obtener las métricas de desempeño.

Se emplean un módulo de evaluación para cada modelo aparte del proceso de entrenamiento, sin realizar un nuevo entrenamiento ni ajuste de los Hiperparámetros de los modelos. Se obtienen las métricas de desempeño de los modelos, generando un archivo EXCEL con las métricas y un archivo de registro con información resumida del proceso.

5.6 Entrenamiento del modelo para el Reconocimiento de entidades nombradas (NER)

Gracias a las facilidades que brinda la librería *spaCy* y su modelo preentrenado de *Deep learning* "*en_core_web_sm*" para el desarrollo del modelo de reconocimiento de entidades nombradas (NER), se realiza un entrenamiento desde cero (*Training from Scratch*) para el componente NER. Adicionalmente, se reutilizan los componentes lingüísticos, "tok2vec" (vectores), "tagger" (etiquetas POS), "parser" (dependencias), "attribute_ruler" (normaliza atributos) y "lemmatizer" (lematización) , transferidos desde el modelo preentrenado *en_core_web_sm*. Desde la perspectiva del *Transfer Learning*, esta estrategia corresponde a un enfoque basado en características (*Feature-Based*), en el que se aprovecha conocimiento lingüístico previamente aprendido para apoyar una nueva tarea de extracción de información.

Usando *spaCy*, *Scikit-learn*, el *Dataset* para la tarea NER y un módulo con las métricas de evaluación, se realizar el proceso de selección, optimización y evaluación del modelo correspondiente al modelo NER, como se describe en la anterior Sub-sección (5.5). Los datos del *Dataset* se dividen en dos grupos, usando una semilla aleatoria ($seed = 42$), en 80 % para entrenamiento y 20 % para validación.

Para el proceso correspondiente al modelo NER se necesita encontrar los hiper-parámetros adecuados: "*Epoch*", "*dropout*", "*batch_size*" y "*learn_rate*", usando como estrategia de optimización de hiper-parámetros (Ciclo de punto 2 de la Sub-sección 5.5) la Búsqueda en cuadrícula (*Grid Search*) y se aplica el *Early Stopping*. Se usa *Early Stopping* con un valor de paciencia, para guardar el mejor entrenamiento de cada combinación. Además, se usaron técnicas complementarias para mejorar el proceso:

- Balanceo de clases en las entidades nombradas, mediante el sobremuestreo de las entidades minoritarias.
- Ajuste progresivo del tamaño del batch, usando la función compounding de spaCy, $compounding(start, stop, compound)$. Donde cada nuevo batch $\approx$ batch anterior $\times$ compound.

Para cada combinación de hiperparámetros evaluada durante el proceso de *Grid Search*, se almacenaron automáticamente las métricas de desempeño y los modelos generados. Los resultados obtenidos fueron consolidados un EXCEL para facilitar la comparación entre configuraciones y la selección de aquella que presentó el mejor desempeño. Adicionalmente, se registró información resumida de cada ejecución con fines de trazabilidad y reproducibilidad del proceso experimental.

La mejor combinación de hiperparámetros encontrada para el modelo NER se presenta en la Tabla 10.

Tabla 10

Configuración del mejor modelo NER, llamado "MODEL_ner".

Hiperparametro	valor
Epoch	26
dropout	0.3
batch_size	[2, 12, 1.01]
learn_rate	0.0005
Paciencia (Early stopping)	8
semilla aleatoria	42

Luego el modelo seleccionado se valida mediante validación cruzada k-Fold convencional, tomando el *Dataset* para la tarea NER y dividiéndolo en K ($K = 5$) sub-grupos que se reutilizan para el entrenamiento y la validación, como se describe en la Sub-sección 3.12.7.2. Nuevamente, usando un módulo se aplican las métricas de evaluación y se obtienen los resultados del proceso de validación, que se guardan en un EXCEL, generando un archivo de registro con información resumida del proceso. Por último se realiza la evaluación independiente (Sección 5.5.1) y se obtiene el desempeño final del modelo.

5.7 Entrenamiento del modelo para la Extracción de relaciones (RE)

Se emplea el modelo preentrenado de SciBERT "$allenai/scibert_scivocab_uncased$" disponible en el *Model Hub* de la plataforma *Hugging Face* (Face, 2026a), el cual se adapta y entrena utilizando la librería *Transformers* de *Hugging Face* (Face, 2026b) aplicando la estrategia de *Fine Tuning* con aprendizaje supervisado para clasificación de relaciones.

El modelo SciBERT preentrenado se inicializa con capas de salida específicas de clasificación y posteriormente se entrena usando el *Dataset* para la tarea RE. Durante el entrenamiento, las secuencias de entrada tokenizadas se procesan en mini-lotes y los parámetros del modelo se actualizaban mediante Retropropagación usando el optimizador AdamW. El script permite realizar el proceso de selección, optimización y evaluación del modelo correspondiente al modelo RE, como se describe en la Sub-sección 5.5. El *Dataset* se divide internamente en dos grupos usando una semilla aleatoria ($seed = 42$), 80 % para entrenamiento y 20 % para validación.

En el proceso correspondiente al modelo RE se necesita encontrar los hiper-parámetros adecuados: "*Epoch*", "*dropout*", "*Effective batch_size*", "*learn_rate*", "*Weight_decay*", "*Freeze_Layers*" (numero de capas congeladas), etc, usando como estrategia de optimización de Hiperparametros (Ciclo de punto 2 de la Sub-sección 5.5) la Búsqueda en cuadrícula (*Grid Search*) y utilizando la biblioteca *Transformers* de *Hugging Face* (Face, 2026b) que tiene integrado el *Early stopping* mediante el mecanismo de *callback* incorporado para interrumpir el entrenamiento, a través del monitoreo del macro-F1 para evitar sobreajustes.

En cada combinación evaluada mediante *Grid Search*, el modelo se evalúa al final de cada época y se almacenan puntos de control. Al finalizar el entrenamiento, se selecciona y recarga automáticamente el punto de control que presenta la mayor puntuación macro-F1 sobre el conjunto de evaluación. Adicionalmente, se registran las métricas obtenidas durante el entrenamiento y la evaluación, así como la evolución de la función de pérdida y del macro-F1, con el fin de facilitar la comparación entre configuraciones y la selección del modelo con mejor desempeño.

Para comparar las métricas de evaluación obtenidas en cada carpeta generada, se desarrolla un módulo que lee los hiperparámetros y las métricas de evaluación, y los guarda en un EXCEL. Al comparar los modelos se encuentra la mejor combinación de hiperparámetros para el modelo RE, como se ve en la Tabla 11.

Tabla 11

Configuración del mejor modelo RE, llamado "MODEL_re".

Hiperparametro	**valor**	**Hiperparametro**	**valor**
Longitud máxima de secuencia	512	Weight decay	0.01
Freeze_Layers	0	Per-device batch size	4
Learning rate	3×10^{-5}	Gradient accumulation steps	4
Dropout	0.1	Effective batch size	16
Paciencia (Early stopping)	3	Máximo numero de Epochs	12
Límite (Early stopping)	2×10^{-3}	Optimizador	AdamW
Semilla aleatoria	42		

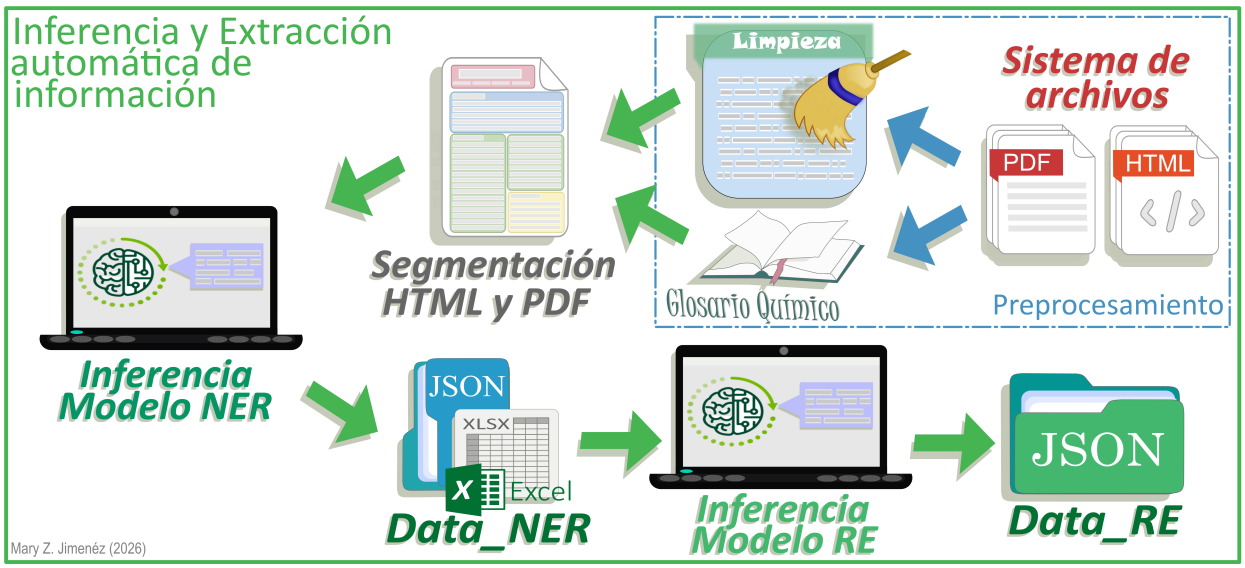
Nota. Effective batch size = Per-device batch size × Gradient accumulation steps

Luego el modelo seleccionado se valida mediante validación cruzada k-Fold estratificada. Se toma el *Dataset* para la tarea RE y se divide en K ($K = 10$) sub-grupos con las clases distribuidas igual en cada sub-grupo, que se reutilizan para el entrenamiento y la validación, como se describe en la Sub-sección 3.12.7.2. Los valores de las métricas obtenidas de la validación se guardan en un EXCEL y en un archivo de registro con información resumida del proceso. Por último se aplica el proceso de la Sección 5.5.1, la evaluación independiente y se obtiene el desempeño final del modelo.

5.8 Inferencia y extracción automática de información

Figura 21

Representación conceptual del proceso de inferencia y extracción automática de información.



Finalizada la fase de entrenamiento y evaluación, los modelos finales (*MODEL_ner* y *MODEL_re*) se emplean en fase de inferencia, para la extracción automática de la información de la totalidad de los 1308 artículos OA guardados en la carpeta *DATA_TRAIN* provenientes de la base de datos manual descrita en el artículo Jacobsson et al. (2022). Las relaciones obtenidas entre entidades son a nivel de ventanas de 2 oraciones.

5.8.1 Extracción del texto y Reconocimiento de entidades nombradas (NER)

Una vez entrenado, el modelo *MODEL_ner* se aplica en fase de inferencia sobre el conjunto completo de artículos para identificar automáticamente las entidades de interés. Previamente, el contenido textual de cada documento es extraído, limpiado y segmentado utilizando los proce-

dimientos descritos en la Sección 5.4.1. Posteriormente, la información obtenida se organiza de forma estructurada según las secciones del artículo y se representa mediante ventanas de dos oraciones junto con las entidades identificadas en cada una de ellas y se guardan en un JSON. Esta representación intermedia permite preservar el contexto local de las entidades y constituye la base para las etapas posteriores de extracción de relaciones y post-procesamiento. La estructura de representación utilizada se ilustra en el Apéndice E.3 y en el ejemplo de la Figura 22.

Figura 22

Ejemplo de la estructura de representación utilizada en los resultados de la inferencia del modelo NER.


```
"Abstract": [
    {"texto_Abstract": "We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar .....",
    "ventanas": ["Ventana 0-1","Ventana 1-2",.....]},
    ["We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar cells (SCs) with a room
    temperature (RT) sputtered ZnO electron transport layer by adding fullerene (C60) interlayer. ZnO exhibits
    a better matched conduction band level with perovskite and Al work function and around energy offset of 2.2
    eV between highest occupied molecular orbital level of CH3NH3PbI3 perovskite and valance band level of ZnO.",
    {"entities": [["CH3NH3PbI3",49,59,"PTF","PTF_0"],["perovskite",60,70,"PTF","PTF_1"],
            ["room temperature",102,118,"ENMTDS","ENMTDS_2"],["ZnO",134,137,"ETL","ETL_3"],
            ["C60",184,187,"ETL","ETL_4"],["ZnO",201,204,"ETL","ETL_5"],["perovskite",258,268,"PTF","PTF_6"],
            ["CH3NH3PbI3",377,387,"PTF","PTF_7"],["perovskite",388,398,"PTF","PTF_8"]]},
    {"entities_seccion": [......]}],
    ["ZnO exhibits a better matched conduction band level with perovskite and Al work function ......",
    {"entities": [["ZnO",0,3,"ETL","ETL_0"],["perovskite",57,67,"PTF","PTF_1"],.....]},
    {"entities_seccion": [["ZnO",201,204,"ETL","ETL_0"],["perovskite",258,268,"PTF","PTF_1"],.....]}],
    ......]
```


Nota. Se muestra la estructura de los datos en formato JSON.

Además de las ventanas de dos oraciones junto a sus entidades identificadas por el modelo NER, se extrae una representación estructurada de las entidades presentes en cada artículo, la cual sirve luego para determinar la eficacia de la extracción NER. De forma complementaria, se identifican y extraen las tablas presentes en los documentos mediante procedimientos específicos para archivos PDF y HTML, aprovechando tanto la estructura del documento como la información de

posicionamiento del contenido. Los detalles de implementación de este procedimiento se describe en el Apéndice D. Finalmente, se registra información resumida de la ejecución con fines de trazabilidad y reproducibilidad del proceso. Todos los archivos generados son guardados en una carpeta *DATA_NER*.

5.8.2 Extracción de relaciones (RE)

Figura 23

Ejemplo de la estructura de representación utilizada en los resultados de la inferencia del modelo RE.


```
"Abstract": [
    {"texto_Abstract": "We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar .....",
    "ventanas": ["Ventana 0-1","Ventana 1-2",.....]},
    ["We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar cells (SCs) with.....",
    {"text_market": "We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar cells
    (SCs) with a room temperature (RT) sputtered ZnO electron transport layer by adding fullerene (<e1>C60</e1>)
     interlayer. ZnO exhibits a better matched conduction band level with perovskite and Al work function and
     around energy offset of 2.2 eV between highest occupied molecular orbital level of <e2>CH3NH3PbI3</e2>
     perovskite and valance band level of ZnO.",
        "relation": "USED_WITH","probability": 0.9993107318878174,
        "entities": [["C60",184,187,"ETL","ETL_4"],["CH3NH3PbI3",377,387,"PTF","PTF_7"]],
        "entities_seccion": [["C60",184,187,"ETL","ETL_4"],["CH3NH3PbI3",377,387,"PTF","PTF_7"]]},
    .....],
    ["ZnO exhibits a better matched conduction band level with perovskite and Al work function ......",
    .....],
    ...]
```


Nota. Se muestra la estructura de los datos en formato JSON.

El modelo *MODEL_re* se emplea en fase de inferencia sobre los archivos de la carpeta *DATA_NER* para convertir los datos e identificar automáticamente las relaciones a nivel de ventanas de dos oraciones. Se hace una discriminación de los pares de entidades para reducir el gasto computacional, al inferir solo sobre pares que están en la lista de relaciones etiquetadas. Se adapta la estructura de información por secciones adicionándole la nueva información y se guarda en un

nuevo JSON. La nueva estructura de representación utilizada se ilustra en el Apéndice E.4 y en el ejemplo de la Figura 23.

Los resultados se guardan en la carpeta *DATA_RE* y se registra información resumida de la ejecución con fines de trazabilidad y reproducibilidad del proceso.

5.9 Post-Procesamiento basado en Ingeniería del Conocimiento

Figura 24

Representación conceptual del post-procesamiento basado en Ingeniería del Conocimiento.



Con la idea de transformar la Extracción de Relaciones a nivel de oración que permite el modelo *MODEL_re*, hacia algo parecido a la extracción de relaciones a nivel de documento, se implementan una serie de técnicas basadas en la Ingeniería del Conocimiento (*Know-ledge Engineering*, KE). Como se describe brevemente en la Sección 3.2, son principalmente enfoques basados en reglas heurísticas y la aplicación del conocimientos específicos de la tarea objetivo.

Para implementar lo que en este trabajo se denomina "*extracción de relaciones a nivel de dispositivo/device*", se utilizan las entidades y relaciones obtenidas en la etapa anterior, los datos

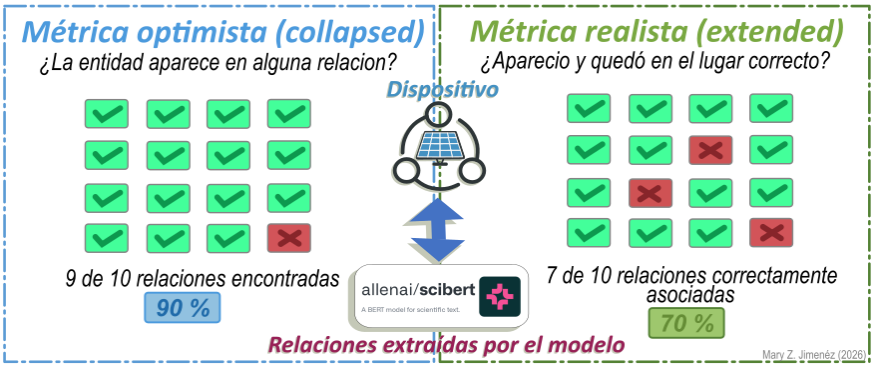
resultados de la fase de inferencia del modelo RE, la extracción de relaciones a nivel de ventanas de 2 oraciones guardados en la carpeta *DATA_RE*.

A partir de esta información, se agrupan las entidades presentes en cada sección del artículo para construir estructuras denominadas *devices*. Estas agrupaciones se generan mediante un conjunto de reglas basadas en expresiones regulares, la posición de las entidades dentro del texto y las relaciones identificadas por el modelo RE. El objetivo es reunir en una misma estructura, las entidades que hacen parte del mismo dispositivo fotovoltaico reportado en el artículo.

Finalmente, sobre cada *device* generado se calculan las métricas complementarias ilustradas en la Figura 25, las cuales permiten evaluar la coherencia y consistencia entre las entidades extraídas y las relaciones inferidas por el modelo RE. El Apéndice F presenta una descripción detallada del procedimiento.

Figura 25

Representación de las métricas complementarias.



Nota. Comparación de la étrica optimista (collapsed) vs métrica realista (extended).

Como resultado se obtiene el *Database*, conformado por un conjunto de archivos JSON almacenados en la carpeta *DATA_RESULTS*. Cada archivo contiene un diccionario anidado, organizado por secciones donde se almacenan los distintos devices reconstruidos, junto con las dos métricas de coherencia utilizadas para su validación y una lista que resume las entidades asociadas. Cada device se compone de entidades agrupadas en torno a tres clases principales de etiquetas: las capas, el desempeño ó los métodos, que permite representar de forma estructurada una celda solar de perovskita. La estructura se ilustra en el Apéndice E.5 y en el ejemplo de la Figura 26.

Figura 26

Ejemplo de la estructura de representación utilizada en los resultados del post-procesamiento.

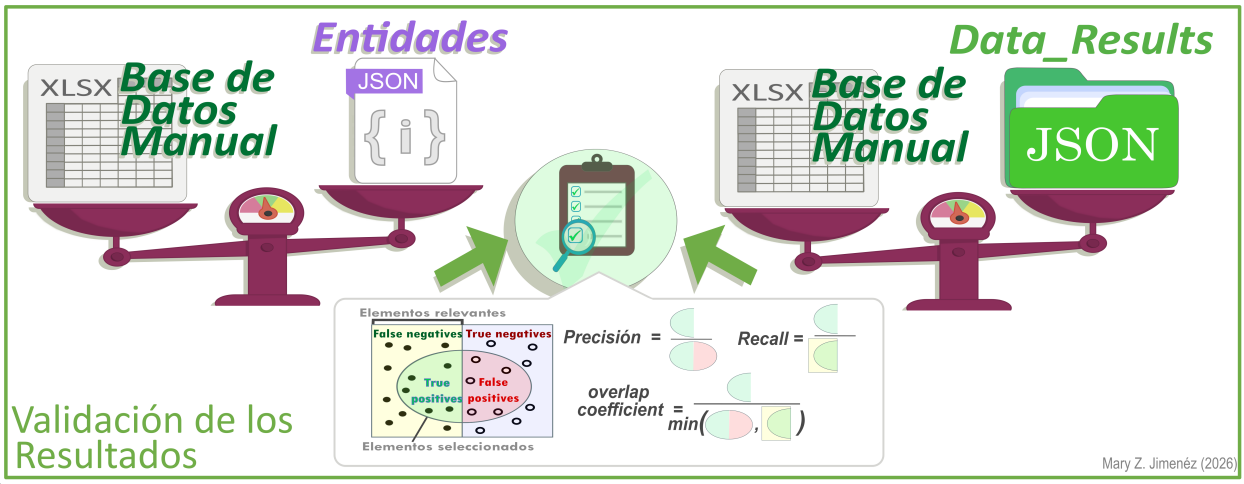

```
"Abstract": [
{"texto_Abstract": "We have demonstrated the performance of inverted CH3NH3PbI3 perovskite-based solar .....",
"AROUND_PERFORMANCE": [{"device_id": 0,
    "structure": "ITO/poly(3,4-ethylenedioxythiophene):poly(styrene-sulfonate) (PEDOT:PSS)/perovskite/C60/
    RT sputtered ZnO/Al",
    "layers": ["C60","ZnO","CH3NH3PbI3","PEDOT:PSS"],"methods": ["sputtering","room temperature"],
    "JSC": "19.41 mA","VOC": "0.91 V","PCE": "10.93%","FF": "",
    "optimistic score": {
        "confirmed_relationships": 14,"model_relationships": 14,"structure_score": 1.0,"entity_support": 1.0,
        "final_confidence": 1.0},
    "realistic score": {
        "confirmed_relationships": 40,"model_relationships": 53,"related_entities": 16,"entities": 16,
        "structure_score": 0.755,"entity_support": 1.0,"distance_support": 0.907,"final_confidence": 0.859},
    "device_summary": {
            "layers": [["ETL_549552",{"text": "C60","type": "ETL","start": 549,"end": 552}],
        ["ETL_693696",{"text": "ZnO","type": "ETL","start": 693,"end": 696}],
        ["PTF_765781",{"text": "CH3NH3PbI3","type": "PTF","start": 765,"end": 781}],
        ["HTL_966975",{"text": "PEDOT:PSS","type": "HTL","start": 966,"end": 975}]],
            "performance": [["JSC_823831",{"text": "19.41 mA","type": "JSC","start": 823,"end": 831}],
        ["VOC_870876",{"text": "0.91 V","type": "VOC","start": 870,"end": 876}],
        ["PCE_13371343",{"text": "10.93%","type": "PCE","start": 1337,"end": 1343}]],
            "methods": [["MTDS_728738",{"text": "sputtering","type": "MTDS","start": 728,"end": 738}],
        ["ENMTDS_102118",{"text": "room temperature","type": "ENMTDS","start": 102,"end": 118}]],
            "structure": [" ITO/poly(3,4-ethylenedioxythiophene):poly(styrene-sulfonate)(PEDOT:PSS)/perovskite
        /C60/RT sputtered ZnO/Al",
        901,1019,[["HTL_966975",{"text": "PEDOT:PSS","type": "HTL","start": 966,"end": 975}],...]}
    },....},.....]
```


Nota. Se muestra la estructura de los datos en formato JSON.

5.10 Validación de los Resultados

Figura 27

Representación conceptual del proceso de validación de los resultados.



Para validar los resultados del proceso de inferencia y post-procesamiento (Sección 5.8 y Sección 5.9), sobre los 1307 artículos OA guardados en la carpeta *DATA_TRAIN* se realizan dos comparaciones, la extracción de entidades de cada artículo y los devices reconstruidos, contra los datos consignados en el *Database* manual (Jacobsson et al., 2022). El resultado de las comparaciones se evalúan mediante tres métricas: precisión, recall y el coeficiente de solapamiento (*overlap coefficient*).

5.10.1 Comparación a nivel de entidad

El módulo trabaja por bloques de artículos, para listar y comparar las entidades extraídas que se guardaron en el JSON *DATANER.json* (Sección 5.8.1) contra el *Database* manual (Jacobsson et al., 2022). Genera la carpeta de resultados *Comparison_ner*. Esta comparación permite

evaluar el desempeño del modelo NER en un entorno realista y respaldar la siguiente comparación, a nivel de dispositivo.

5.10.2 Comparación a nivel de device

El módulo trabaja por bloques de artículos, para listar y comparar los datos del *Database* almacenado en la carpeta *DATA_RESULTS* (Sección 5.8.2) contra el *Database* manual (Jacobsson et al., 2022). Como resultado, se genera la carpeta *Comparison*. Los resultados obtenidos constituyen la principal evidencia experimental para validar el funcionamiento y la capacidad de extracción del *Pipeline* desarrollado en este trabajo.

5.11 Entorno Experimental

Todos los experimentos y el *Pipeline* se realizaron en dos sistemas: en un ordenador personal convencional con *Microsoft Windows* y una *Workstation* universitaria perteneciente al grupo CEMOS. La mayor parte del desarrollo experimental se realizó en el ordenador personal y la *workstation*, disponible únicamente en la etapa final del proyecto, fue utilizada para el entrenamiento del modelo RE (Sección 5.7) y como apoyo en la parte RE de la generación del *database* (Sección 5.8.2). Se empleó una configuración computacional de recursos moderados, representativa de entornos de investigación no especializados. La implementación fue desarrollada en Python utilizando *PyTorch*, *Hugging Face Transformers* y *spaCy*. En la Tabla 12 se ve el resumen los dos sistemas.

Tabla 12

Entornos en los que se experimento y trabajo el Pipeline.

Componente	Especificación
Ordenador personal	
Sistema	Windows 10 Home
Procesador	Intel core i5-4
RAM	12 GB
GPU	GeForce GTX 1050 Ti
Memoria GPU	4 GB
Versión de Python	3.9
librerías	Transformers, spaCy, scikit-learn, Unpywall, Beautiful Soup, Requests, PyMuPDF
Librerías de LLM	PyTorch
Workstation	
Sistema	Ubuntu 24 LTS
Procesador	Intel xeon w7-2
RAM	128 GB
GPU	NVIDIA RTX 4500 Ada Generation
Memoria GPU	24 GB
Versión de Python	3.9
librerías	Transformers
Librerías de LLM	PyTorch

6 Resultados

Con la implementación del *Pipeline* híbrido descrito en el capítulo anterior (Figura 11), centrado en la extracción contextualizada de información sobre celdas solares de Perovskita mediante la agrupación de entidades en dispositivos (*devices*), organizado a nivel de secciones del artículo, se realizaron diversos experimentos cuyos resultados, tanto cuantitativos como cualitativos, se presentan en las siguientes sub-secciones para facilitar su análisis e interpretación.

6.1 Web Scraping para descargar artículos

Del proceso de la Sección 5.1 realizado entre finales de 2024 y principios de 2025 usando el ordenador personal descrito en la Sección 5.11, se obtuvieron 1309 artículos de acceso abierto (*Open Access*, OA) en formato PDF y HTML almacenados en un sistema de archivos en la carpeta *DATA_TRAIN*, a partir de 5107 artículos de celdas solares de Perovskita entre los años 2003 a 2020 del "*The Perovskite Database Project*" (Jacobsson et al., 2022).

De los 5107 artículos iniciales, se determinó automáticamente que 1308 correspondían a documentos OA, mientras que para 5 no fue posible determinar su estado. Mediante Web scarping se descargan de forma semi-automática de 1308 HTMLs, durante 9 horas con 19 minutos utilizando el ordenador personal descrito en la Sección 5.11. Se detecta de forma automática que de estos archivos, 626 eran artículos en formato HTML y 681 eran en realidad paginas con el link para descargar el artículo en formato PDF. Y al revisar los 681 y los 5 a los que antes no se pudo

determinar el estado, se obtienen y descargan manualmente 683 artículos OA en formato PDF. En la Tabla 13 se ve un resumen.

Tabla 13

Resumen del proceso de Selección y Recolección de los datos.

Etapa		Cantidad
Artículos Iniciales		5107
Detección automática de estado:	OA	1308
	No detectados	5
Descargados semi-automáticamente en HTML		1308
Descargados manualmente en PDF		683
Artículos finales:	HML	626
	PDF	683
	TOTAL	1309

La detección automática del formato real del artículo como se describe en la sub-seccion 5.1.2 se realizo sobre la estructura HTML de 20 paginas web de Jornals, listadas a continuación.

Code Listing 6.1. Lista de paginas web.

```
[
        "https://pubs.acs.org",
        "https://onlinelibrary.wiley.com",
        "https://advanced.onlinelibrary.wiley.com",
        "https://pubs.aip.org",
        "https://academic.oup.com",
        "https://www.cell.com",
        "https://pubs.rsc.org",
        "https://www.mdpi.com",
        "https://www.nature.com",
        "https://link.springer.com",
        "https://www.frontiersin.org",
        "https://ieeexplore.ieee.org",
        "https://www.aimspress.com",
        "https://www.sciencedirect.com",
        "https://www.jstage.jst.go.jp",
        "https://iopscience.iop.org",
        "https://www.science.org",
```

```
        "https://www.tandfonline.com",
        "https://royalsocietypublishing.org",
        "https://opg.optica.org/"
]
```

Dejando de lado la descarga manual, que fue necesaria debido al formato en el que se comparten algunos artículos OA, el proceso de descarga en formato HTML usando web scarping alcanzó una tasa de recuperación del 100 %. También, el proceso de detección automático del formato, no solo permitió saber que artículos necesitaban ser descargados manualmente en PDF, sino que luego ayudo a segmentar el texto y automatizar la construcción de los datos etiquetados (*Dataset*), reduciendo intervención manual y facilitando el proceso de desarrollo de los modelos.

6.2 Construcción del conjunto de datos etiquetados (Dataset) para los modelos NER y RE

Al aplicar el preproceso (Sección 5.2) y el proceso de generación del los datos etiquetados de la Sección 5.4 sobre el sistema de archivos en la carpeta *DATA_TRAIN*, se obtiene un archivo JSON con los datos etiquetados (*Dataset*) para el entrenamiento del modelo NER de la Sección 5.6. La parte automática de la generación de los datos etiquetados necesito de 1 hora y 22 minutos, utilizando el ordenador personal descrito en la Sección 5.11. El *Dataset* contiene 4082 oraciones con 14014 entidades etiquetadas, distribuido como se ve en la Tabla 14.

Para mejorar el proceso de entrenamiento del modelo NER, fue necesario balancear las clases de entidades del *Dataset* mediante sobremuestreo de las muestras correspondientes a etiquetas poco frecuentes. Inicialmente, el *Dataset* se dividió usando una semilla aleatoria ($seed = 42$), en 80 % para entrenamiento y 20 % para validación, obteniéndose 3265 oraciones y 11239 entidades etiquetadas para el conjunto de entrenamiento, y 817 oraciones y 2775 entidades etiquetadas para

el conjunto de validación. Luego, las etiquetas con una frecuencia igual o inferior a 200 entidades del conjunto de entrenamiento fueron sobremuestreadas mediante su duplicación, con un máximo de dos duplicaciones por oración. La nueva distribución de los archivos luego de balancearlos se ve en la Tabla 15.

Tabla 14

Distribución del Dataset para la tarea NER.

Clase	**Conteo**	**%**	**Clase**	**Conteo**	**%**
PTF	2527	18.03	ENMTDS	451	3.22
ETL	1995	14.24	JSC	399	2.85
MTDS	1983	14.15	FF	373	2.66
NMTDS	1796	12.82	NVOC	252	1.80
HTL	1604	11.45	VMTDS	243	1.73
PCE	753	5.37	NJSC	242	1.73
NPCE	733	5.23	NFF	194	1.38
VOC	469	3.35			

Nota. Dataset guardado en el archivo *TRAINHTMLL_RS*1*_NER_RS.json*.

Tabla 15

Distribución del Dataset para la tarea NER balanceados.

Clase	**Entrenamiento**	**Validación**	**Clase**	**Entrenamiento**	**Validación**
PTF	2195	488	ENMTDS	746	91
ETL	1683	412	JSC	685	75
MTDS	1623	396	FF	594	74
NMTDS	1458	366	NVOC	503	47
HTL	1356	330	VMTDS	474	44
PCE	913	144	NJSC	363	36
NPCE	911	140	NFF	223	36
VOC	772	96			
Total entidades	14499	2775			
Oraciones	3635	817			

Para la validación cruzada por K-Fold, se utiliza el *Dataset* original (Figura 14) y también se aplica el balanceo al conjunto de entrenamiento. En cada iteración del K-Fold, primero se realiza la partición de los datos en entrenamiento y validación. Posteriormente, el sobremuestreo se aplica al conjunto de entrenamiento, mientras que el conjunto de validación permanece sin modificaciones.

Luego para realizar el entrenamiento del modelo RE, como se describió en la Sección 5.7, se transformo el *Dataset*, convirtiendo su formato, generando los pares entidades-relación y corrigiendo manualmente, hasta obtener el archivo JSON con los datos etiquetados (*Dataset*). El archivo contiene 5900 pares entidades-relación con 662 oraciones únicas, distribuido como se ve en la Tabla 16. Se divide internamente en dos grupos usando una semilla aleatoria ($seed = 42$) y división estratificada, 80 % para entrenamiento y 20 % para validación.

Tabla 16

Distribución del Dataset de los datos etiquetados para la tarea RE.

Clase	Conteo	%
NO_RELATION	4019	68.118644
USED_WITH	571	9.677966
PERFORMED_UNDER_CONDITION	502	8.508475
PREPARED_WITH	404	6.847458
HAS_VALUE	404	6.847458

Nota. Dataset guardado en el archivo $TRAINHTMLL_RS1_L_RE_C_llm.json$.

6.2.1 Construcción del Dataset para la evaluación (Test) independiente de los modelos NER y RE

Para la evaluación (*Test*) independiente del modelo NER como se describe en la Sección 5.5.1, nuevamente se aplica el preproceso (Sección 5.2) y el proceso de generación del los datos etiquetados de la Sección 5.4.4 sobre el sistema de archivos de la carpeta $DATA_TEST$ y se obtiene un archivo JSON con los datos etiquetados (*Dataset*) de evaluación (*Test*). La parte automática de la generación de los datos etiquetados necesito de 50 minutos, utilizando el ordenador personal descrito en la Sección 5.11. El *Dataset* contiene 923 oraciones con 3450 entidades etiquetadas, distribuido como se ve en la Tabla 17.

Tabla 17

Distribución del Dataset usado en la evaluación independiente del modelo NER.

Clase	**Conteo**	**%**	**Clase**	**Conteo**	**%**
PTF	569	16.49	JSC	107	3.10
ETL	423	12.26	ENMTDS	105	3.04
MTDS	404	11.71	NVOC	93	2.70
NPCE	366	10.61	FF	88	2.55
NMTDS	347	10.06	NJSC	84	2.43
HTL	333	9.65	NFF	73	2.12
PCE	324	9.39	VMTDS	25	0.72
VOC	109	3.16			

Nota. Dataset guardado en el archivo *Test_NER_RS_L.json*.

Luego para realizar la evaluación (*Test*) independiente del modelo RE, como se describió en la Sección 5.5.1, se transformo el *Dataset* de *Test*, convirtiendo su formato, generando los pares entidades-relación y corrigiendo manualmente, hasta obtener el archivo JSON con los datos eti-

quetados. El archivo contiene 3934 pares entidades-relación con 562 oraciones únicas, distribuido como se ve en la Tabla 18.

Tabla 18

Distribución del Dataset usado en la evaluación independiente del modelo RE.

Clase	Conteo	%
NO_RELATION	2559	65.048297
PERFORMED_UNDER_CONDITION	440	11.184545
HAS_VALUE	386	9.811896
USED_WITH	331	8.413828
PREPARED_WITH	218	5.541434

Nota. Dataset guardado en el archivo *Test_RE_C_llm.json*.

6.3 El modelo NER

El proceso de selección, optimización y validación del modelo se realizo con Búsqueda en cuadrícula (*Grid Search*) usando los valores de Hiperparametros de, *epoch* máximo 50, semilla aleatoria= 42, $dropout = [0.2, 0.3, 0.4, 0.35]$, *batch_size* con ajuste progresivo $[2, 12, 1.01]$ y $learn_rate = [0.001, 0.0005, 0.002, 0.0007]$ tal como se describe en la Sección 5.6. Con este *Grid Search* se generan 16 combinaciones como se ven en la Tabla 19 y en el la Figura 35 del Apéndice G.1. El proceso tardo en total 2 horas y 44 minutos utilizando el ordenador personal descrito en la Sección 5.11.

Tabla 19

Resumen de las 16 combinaciones evaluadas para el modelo NER.

Combinacion	dropout	lr	iteraciones	micro-F1	Tiempo
1	0.2	0.001	10	0.948201	0:07:38.585358
2	0.2	0.0005	6	0.948837	0:06:02.705295
3	0.2	0.002	13	0.949055	0:08:49.735410
4	0.2	0.0007	15	0.95026	0:09:40.267873
5	0.3	0.001	18	0.949723	0:10:46.958096
6	**0.3**	**0.0005**	**26**	**0.950837**	**0:13:46.794602**
7	0.3	0.002	14	0.947594	0:08:58.938113
8	0.3	0.0007	12	0.947898	0:08:08.485603
9	0.4	0.001	18	0.952569	0:10:27.378323
10	0.4	0.0005	18	0.948325	0:10:29.768159
11	0.4	0.002	24	0.950766	0:12:46.123259
12	0.4	0.0007	16	0.95017	0:09:39.699800
13	0.35	0.001	20	0.951904	0:11:11.689299
14	0.35	0.0005	26	0.949455	0:13:29.907499
15	0.35	0.002	17	0.949378	0:10:01.661899
16	0.35	0.0007	16	0.950449	0:09:37.307279

Nota. Las combinaciones corresponden a la búsqueda de hiperparámetros realizada mediante *Grid Search*.

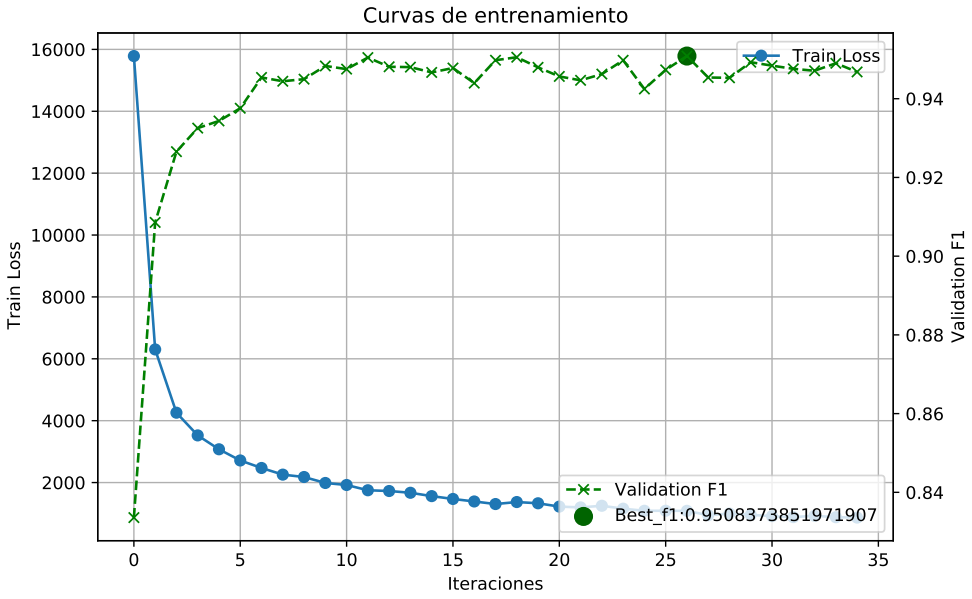
Gracias a la Tabla 19 y las gráficas de la Figura 35 del Apéndice G.1 se escoge como la mejor combinación de hiperparámetros para el modelo, la combinación numero 6. Para la selección se tiene en cuenta la suavidad de la curva de entrenamiento y el valor del F1-score-micro vs el tiempo de entrenamiento. Con los hiperparámetros seleccionados se realiza otro entrenamiento obteniendo el modelo *MODEL_ner*:

- *Con Balanceo de entidades*
- *epoch* máximo 50

- *dropout* $= 0.3$
- *Batch size* con ajuste progresivo que empieza en 2 hasta 12 y aumenta en 1.01 veces cada epoch ([2, 12, 1.01])
- *Paciencia (Early_stopping)*$= 8$
- *Learning rate* $= 5e-04$
- *Semilla aleatoria* $= 42$

Figura 28

Gráfica de las curvas de entrenamiento del modelo NER (MODEL_ner) seleccionado.



Nota. Se muestran las curvas F1 vs Iteración y pérdida (Loss) vs Iteración.

Todos los archivos generados en esta Sección se almacenan en la carpeta *Experiments_ner*.

6.3.1 Evaluación del modelo sobre el conjunto de validación

El modelo *MODEL_ner* es validado usando las 817 oraciones con las 2775 entidades etiquetadas del conjunto de validacón. Las métricas se reportan a nivel global mediante micro-

average, y a nivel por clase con macro-average. Los resultados de evaluación guardadas en el EXCEL *Model_ner_Validation.xlsx*. En la Tabla 20 se ve el resumen de las métricas y en la Tabla 21 se pueden observar las métricas por entidad del modelo.

Tabla 20

Métricas del modelo MODEL_ner en la evaluación sobre el conjunto de validación.

	Precision	**Recall**	**F1-score**
Micro-average (global)			
Micro-average	0.9503	0.9514	0.9508
Macro-average (por clase)			
Macro-mean	0.9335	0.9316	0.9268
Std	0.1923	0.1933	0.1854
Min	0	0	0
25 %	1	1	0.924451
50 %	1	1	1
75 %	1	1	1
Max	1	1	1
var	0.0369	0.0373	0.0343

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average calcula la métrica por clase y luego promedia sin ponderar, lo que permite evaluar el desempeño equilibrado entre entidades.

Se observa en la Tabla 20 que las métricas *macro-average* lograron un valor cercano y ligeramente más alto que las *micro-average*, evidencia de la estabilidad lograda por modelo en el proceso de entrenamiento, generada gracias al balance de clases que se aplico a los datos de entrenamiento y evaluación. Estos resultados corresponden a la evaluación empleada durante el ajuste y selección del modelo.

Tabla 21

Métricas promedio por entidad del modelo MODEL_ner sobre el conjunto de validación.

Entidad	N	Precision	std	Recall	std	F1-score	std
ETL	227	0.980	0.136	0.980	0.136	0.979	0.135
HTL	194	0.938	0.229	0.938	0.229	0.935	0.227
PTF	313	0.941	0.222	0.938	0.224	0.936	0.220
MTDS	239	0.925	0.247	0.929	0.242	0.924	0.242
VMTDS	35	0.500	0.5	0.486	0.492	0.486	0.487
NMTDS	140	0.932	0.242	0.907	0.249	0.915	0.242
ENMTDS	88	0.801	0.398	0.782	0.398	0.787	0.394
PCE	119	0.945	0.214	0.943	0.215	0.939	0.211
VOC	76	0.987	0.115	0.987	0.115	0.987	0.115
JSC	62	0.954	0.191	0.968	0.178	0.959	0.183
FF	57	1	0	1	0	1	0
NPCE	96	0.845	0.347	0.843	0.349	0.839	0.344
NVOC	44	0.807	0.392	0.807	0.392	0.803	0.389
NJSC	36	0.898	0.296	0.880	0.297	0.879	0.291
NFF	33	0.727	0.452	0.712	0.451	0.717	0.450

Nota. N es numero entidades que identifico el modelo para cada etiqueta.

6.3.2 Validación cruzada del modelo

Aplicar la validación cruzada por K-Fold al modelo $MODEL_ner$ con $k = 5$, tarda 59 minutos y sus resultados se guardan en el EXCEL $MODEL_ner_Kfold - Validation.xlsx$. Los valores promedio de los resultados, se ven en la Tabla 22 y las graficas de la Figura 36 del Apéndice G.2.

Tabla 22

Métricas promedio de la validación cruzada por k-Fold del modelo MODEL_ner, con $K = 5$.

	Micro-average (global)			Macro-average (por clase)		
	ents_Precision	**ents_Recall**	**ents_F1**	**ents_Precision**	**ents_Recall**	**ents_F1**
mean	0.9381	0.9538	0.9458	0.9219	0.9278	0.9197
std	0.0106	0.0086	0.0045	0.2020	0.1992	0.1945
min	0.9215	0.9409	0.9402	0	0	0
25 %	0.9358	0.9496	0.9426	0.9833	1	0.9139
50 %	0.9401	0.9566	0.9454	1	1	1
75 %	0.9430	0.9598	0.9498	1	1	1
max	0.9499	0.9620	0.9509	1	1	1
var	1.13E-04	7.32888E-05	2.06575E-05	0.0409	0.0397	0.0379

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average calcula la métrica por clase y luego promedia sin ponderar, lo que permite evaluar el desempeño equilibrado entre entidades.

Se observa en la validación cruzada K-fold que las métricas micro y macro (Tabla 22) presentan un desempeño similar al obtenido sobre el conjunto de validación utilizado durante el desarrollo del modelo (Tabla 20). la Tabla 22 también muestra una variabilidad reducida en las métricas micro, mientras las métricas macro presentaron una mayor variación que puede estar relacionado con la distribución de entidades entre los folds.

6.3.3 Evaluación (Test) independiente del modelo

Una vez entrenado, optimizado y validado el modelo final, *MODEL_ner*, se realiza una evaluación sobre el conjunto de Test independiente con 923 oraciones y 3450 entidades etiquetadas. Las métricas se reportan a nivel global mediante micro-average, y a nivel por clase con macro-average. Adicionalmente, se presentan resultados desagregados por tipo de entidad para analizar el

comportamiento del modelo en cada clase. Los resultados de evaluación guardadas en el EXCEL *Model_ner_Evaluation.xlsx*. En la Tabla 23 se ve el resumen de las métricas y en la Tabla 24 se puede observar el desempeño por entidad del modelo.

Tabla 23

Desempeño del modelo MODEL_ner sobre un Test independiente.

	Precision	**Recall**	**F1-score**
Micro-average (global)			
Micro-average	0.9084	0.9026	0.9055
Macro-average (por clase)			
Macro-mean	0.8841	0.8810	0.8730
Std	0.2427	0.2423	0.2330
Min	0	0	0
25 %	0.8750	0.8571	0.8550
50 %	1	1	1
75 %	1	1	1
Max	1	1	1
var	0.0589	0.0586	0.0542

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average calcula la métrica por clase y luego promedia sin ponderar, lo que permite evaluar el desempeño equilibrado entre entidades.

Las métricas del conjunto *test* independiente y la validación a pesar de usar dos conjuntos diferentes de datos, presentan disminución de sus métricas en las mismas etiquetas de entidades. En las Tablas 21 y 24, se nota que las entidades etiquetas como "$VMTDS$", "$ENMTDS$" y "NFF" presentan las menores métricas macro. En el caso de las etiquetas "$VMTDS$" y "$ENMTDS$" es debido a la alta variabilidad léxica de las etiquetas, donde se representan los valores de los parámetros y el entorno de los métodos de síntesis, respectivamente, como son, "$VMTDS$": "*temperature*",

"*time*", "*speed*", "*pressure*", "*viscosity*", "*rate*", y"*ramp*", "*ENMTDS*": "*O*2", "*nitrogen*","*low* − *temperature*" y "*N*2 *atmosphere*". Por otro lado la etiqueta "*NFF*" presenta para el sistema de identificación del modelo un reto debido a su falta de unidades en la mayoría de casos, lo que lleva a una identificación mayor de falsos positivos respecto a otras entidades.

Tabla 24

Desempeño promedio por entidad del modelo MODEL_ner sobre un Test independiente.

Entidad	**N**	**Precision**	**std**	**Recall**	**std**	**F1-score**	**std**
ETL	287	0.935	0.235	0.918	0.250	0.921	0.239
HTL	193	0.842	0.343	0.819	0.352	0.819	0.341
PTF	432	0.841	0.340	0.858	0.335	0.842	0.332
MTDS	297	0.878	0.319	0.878	0.319	0.875	0.318
VMTDS	33	0.545	0.506	0.545	0.506	0.545	0.506
NMTDS	179	0.855	0.336	0.836	0.341	0.839	0.334
ENMTDS	113	0.743	0.434	0.739	0.434	0.737	0.431
PCE	263	0.894	0.294	0.904	0.288	0.895	0.288
VOC	80	0.981	0.124	0.988	0.112	0.983	0.117
JSC	74	0.926	0.258	0.905	0.270	0.912	0.262
FF	62	0.952	0.216	0.944	0.224	0.946	0.219
NPCE	261	0.930	0.235	0.914	0.247	0.917	0.238
NVOC	72	0.819	0.387	0.771	0.393	0.787	0.385
NJSC	71	0.873	0.335	0.864	0.341	0.866	0.338
NFF	59	0.446	0.489	0.412	0.466	0.424	0.470

Nota. N es numero entidades que identifico el modelo para cada etiqueta.

En general el conjunto *test* independiente (Tabla 23) muestra el mismo comportamiento general identificado durante la validación (Tabla 20), aunque con una disminución en los valores de las métricas. Este comportamiento indica que el modelo mantiene las tendencias observadas

durante su desarrollo al enfrentarse a datos no utilizados previamente, pero presenta una reducción en su desempeño.

6.4 Selección de modelo RE

Para la selección del modelo preentrenado a usar para la tarea RE se realizaron dos tipos diferentes de pruebas: comparación de arquitecturas y comparación de modelos. Todos los archivos generados en esta Sección se almacenan en la carpeta $Experiments_re$.

6.4.1 Comparación de Arquitecturas

Se comparan los dos enfoques más utilizados de la estrategia del *Transfer Learning* y se usa el ordenador personal descrito en la Sección 5.11 para desarrollarlos.

6.4.1.1 Basado en Características (Feature based). Se usa un modelo LLM solo como un generador de *embeddings* de las entidades y se pasan a un clasificador (como una red neuronal simple o SVM).

6.4.1.1.1 Regresión logística como clasificador. Se usa el modelo preentrenado liviano "$all-MiniLM-L6-v2$" y las funciones *LogisticRegression* y *GridSearchCV* de la librería *scikit-learn*, para obtener el mejor resultado de la estructura. Los resultados obtenidos se almacenan en la carpeta R_f_ext, incluyendo el resultado de métricas macro de evaluación: $Precision = 0.58$, $Recall = 0.73$ y $F1-Score = 0.63$.

6.4.1.1.2 Máquina de Vectores de Soporte (SVM) como clasificador. Se usa el modelo preentrenado de SciBERT "$allenai/scibert_scivocab_uncased$" y las funciones *SVC* y *GridSearchCV* de la librería *scikit-learn*, para obtener el mejor resultado de la estructura. Los resultados obtenidos

se almacenan en la carpeta *R_llm_svm*, incluyendo el resultado de métricas macro de evaluación: $Precision = 0.77$, $Recall = 0.91$ y $F1 - Score = 0.83$.

6.4.1.2 Ajuste Fino (Fine Tuning). Se ajusta el modelo de SciBERT "$allenai/scibert_scivocab_uncased$" incluyendo sus capas internas y se agrega una capa de clasificación al final para predecir las relaciones. Con el proceso que corresponde al entrenamiento descrito en la Sección 5.7 sin hacer la optimización de hiperparámetros, limitando los $Epochs$ a máximo 3 y solo entrenando las 6 ultimas capas del modelo.

- *GPU = GeForce GTX 1050 Ti de 4GB*
- *Longitud máxima de secuencia (max_length) = 128*
- *Número máximo de Epochs = 3*
- $Freeze_Layers = 6$
- *Per-device batch size* $= 4$
- *Gradient accumulation steps* $= 8$
- *Effective batch size* (*Per-device batch size* $\times$ *Gradient accumulation steps*) $= 32$
- *Learning rate* $= 3 \times 10^{-5}$
- *Weight decay* $= 0.01$

Los resultados obtenidos se almacenan en la carpeta R_fine_tuning, incluyendo el resultado de métricas macro de evaluación: $Precision = 0.87$, $Recall = 0.88$ y $F1 - Score = 0.88$. Todos los archivos generados en esta Sección se almacenan en la carpeta $Experiments_Select_re$.

6.4.2 Comparación de Modelos

Usando el sistema de la Workstation descrita en la Sección 5.11 se comparan diferentes modelos preentrenados que tienen una arquitectura y un formato de ingreso de datos etiquetados similar, de tal manera que se puede aplicar el proceso de adaptación descrito en la Sección 5.7 sin mayores modificaciones. En consecuencia, la comparación se limitó a modelos basados en trans-

formadores disponibles en *Hugging Face's*, seleccionados para representar diferentes tamaños, estrategias de preentrenamiento y grados de especialización de dominio. Este enfoque permitió realizar el ajuste fino y la evaluación bajo un procedimiento experimental homogéneo. Por esta razón, no se incluyeron herramientas que emplean flujos de entrenamiento diferentes, como SpaCy, ya que su incorporación habría requerido una metodología de adaptación distinta y hubiera dificultado la comparación directa de los resultados. La lista de modelos se ve a continuación.

Code Listing 6.2. Lista de modelos preentrenados.

```
[
"allenai/scibert_scivocab_uncased",     # dominio
"allenai/scibert_scivocab_cased",       # dominio + mayusculas
"bert-base-uncased",                    # generalista
"albert-base-v2",                       # comprimido
"prajjwal1/bert-tiny",                  # extremo pequeno
"roberta-base",                         # arquitectura distinta
"bert-large-uncased",                   # grande clasico
"roberta-large",                        # grande moderno
"google/electra-base-discriminator"     # arquitectura alternativa
]
```

Se entrenan los modelos preentrenados de la lista con lo mismos hiperparámetros sin optimizarlos, y se obtienen las métricas macro de evaluación para de cada modelo, como se ve en la Tabla 25.

- *GPU = NVIDIA RTX 4500 Ada Generation de 24 GB*
- *Longitud máxima de secuencia (max_length) = 128*
- *Número máximo de Epochs = 12*
- *Freeze_Layers* $= 3$
- *Per-device batch size* $= 4$
- *Gradient accumulation steps* $= 4$
- *Effective batch size* (*Per-device batch size* $\times$ *Gradient accumulation steps*) $= 16$
- *Learning rate* $= 3 \times 10^{-5}$
- *Weight decay* $= 0.01$
- *Semilla aleatoria* $= 42$

Tabla 25

Resumen de las Métricas de evaluación macro-average para la lista de modelos preentrenados.

Modelo	F1-score	Recall	Precision	Accuracy	Tiempos
albert-base-v2	0.904716	0.897778	0.912987	0.933898	0:08:13.939689
allenai_scibert _scivocab_cased	0.927697	0.931415	0.924415	0.949153	0:11:01.198952
allenai_scibert _scivocab_uncased	**0.926982**	**0.938618**	**0.916197**	**0.948305**	**0:11:00.255693**
bert-base-uncased	0.914767	0.920168	0.909707	0.939831	0:11:20.285227
bert-large-uncased	0.908595	0.897264	0.921882	0.936441	0:24:12.671295
google_electra-base-discriminator	0.921754	0.916923	0.927595	0.945763	0:10:56.170115
prajjwal1_bert-tiny	0.370309	0.373273	0.510442	0.705932	0:02:57.067626
roberta-base	0.922206	0.919689	0.925042	0.945763	0:11:04.809802
roberta-large	0.932143	0.917956	0.947425	0.95339	0:25:19.589999

Nota. El macro-average calcula la métrica por clase y luego promedia sin ponderar, lo que permite evaluar el desempeño equilibrado entre entidades.

6.4.3 Análisis

En la comparación de arquitecturas, se puede observar la mayor estabilidad y mejores valores en las métricas de evaluación para la arquitectura de *Fine Tuning* en comparación con las dos arquitecturas basadas en *Feature based*. Esto resulta particularmente relevante, considerando que la arquitectura de *Fine Tuning* no fue sometida al proceso de selección, optimización y evaluación del modelo (Sub-sección 5.5) para obtener el mejor rendimiento, a diferencia de las otras dos arquitecturas evaluadas.

En la comparación de Modelos preentrenados los resultados muestran, que la adaptación del modelo "*allenai/scibert_scivocab_uncased*", presenta las métricas de evaluación promedio más altas entre los PLMs evaluados, con además tiempos de entrenamiento moderados. Esto es importante considerando que es una arquitectura de menor tamaño en comparación a otras alternativas que obtuvieron valores de métricas de evaluación cercanas, lo que evidencia que tiene la mejor relación entre desempeño y costo computacional.

6.5 El modelo RE

Usando la Workstation descrita en la Sección 5.11 se realizaron diferentes experimentos en el proceso de selección, optimización y evaluación del modelo usando con Búsqueda en cuadrícula (*Grid Search*) tal como se describe en la Sección 5.7. Todos los archivos generados en esta Sección se almacenan en la carpeta $Experiments_re$.

6.5.1 Matriz de hiperparámetros

Se usan los valores de hiperparámetros:

- *GPU = NVIDIA RTX 4500 Ada Generation de 24 GB*
- *Longitud máxima de secuencia (max_length) = 512*
- *Número máximo de Epochs = 12*
- $Freeze_Layers = [0,3,6]$
- *epoch* máximo 12
- $dropout = [0.1, 0.2, 0.3]$
- *Per-device batch size* $= [4,12]$
- *Gradient accumulation steps* $= 4$
- *Paciencia (Early_stopping)*$= 3$
- *Límite(Early_stopping)* $= 2e-03$
- *Learning rate* $= [2e-05, 3e-05, 5e-05]$
- *Weight decay* $= [0.01, 0.05]$

- *Semilla aleatoria* $= 42$

Al variar cinco hiperparámetros de los diez, el *Grid Search* genera 108 combinaciones como se ve en la Tabla 41 del Apéndice H, donde se observan las métricas macro de evaluación, una calculada con todas las relaciones (Macro-average_all) y otra sin la clase *NO_RELATION* (Macro-average_Rel).

Observando la Tabla se deciden seleccionar los hiperparámetros de la combinación 72 con: *Per-device batch size* $= 4$, $learn_rate = 3e-05$,$weight_decay = 0.01$, $dropout = 0.1$, *Freeze_Layers* $= 0$ y un tiempo de entrenamiento de 10 minutos con 53 segundos.

6.5.2 Variación de la semilla aleatoria

Usando los hiperparámetros seleccionados, se varia la semilla aleatoria con 5 valores $[42, 123, 2025, 7, 999]$. Se obtiene la Tabla 26.

Tabla 26

Resumen de las Métricas de evaluación Macro-average_all para diferentes semillas aleatorias.

#	Semilla	F1-score	Recall	Precision	Accuracy	Tiempos
0	42	0.92698217	0.93861798	0.91619702	0.94830508	0:10:21.220157
1	123	0.93162215	0.93635613	0.92715819	0.95169492	0:10:11.701417
2	2025	0.92600139	0.93954997	0.91348833	0.94661017	0:11:45.079556
3	7	0.92800709	0.91668331	0.94072852	0.95	0:11:01.798114
4	999	0.92382119	0.92432105	0.92427583	0.94576271	0:09:32.663973
average		0.927286798	0.931105688	0.92436958	0.948474576	

Nota. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar.

Los resultados obtenidos bajo diferentes valores de inicialización (semilla) aleatoria tienen variaciones pequeñas, entre 0.92 y 0.94. Con esto se puede decir que el modelo para los hiperparámetros seleccionados tiene un comportamiento estable en el proceso de entrenamiento y una baja sensibilidad a la variabilidad introducida por la semilla aleatoria.

6.5.3 Uso del Balanceo de clases

Se prueba la aplicación de balanceo por sobremuestreo de las clases de relaciones. Se modifica el *Dataset* para la tarea RE, pasando de la distribución de la Tabla 16 a tener la distribución balanceada de la Tabla 27.

Tabla 27

Distribución balanceada por sobremuestreo del Dataset para la tarea RE.

Clase	Conteo	Peso en el entrenamiento (Weight)
NO_RELATION	3215	0.1391
USED_WITH	457	0.9787
PERFORMED_UNDER_CONDITION	402	1.1126
PREPARED_WITH	323	1.3848
HAS_VALUE	323	1.3848

Nota. Dataset guardado en $TRAINHTMLL_RS1_L_RE_C_llm.json$.

Tabla 28

Comparación de las Métricas de evaluación del modelo RE, Macro-average_all, para el entrenamiento con y sin balanceo.

Balanceo	F1-score	Recall	Precision	Accuracy	Tiempos
CON	0.9170	0.9438	0.8965	0.9356	0:08:46.826272
SIN	0.92698	0.9386	0.9161	0.9483	0:10:21.220157

Nota. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar.

Balanceado el *Dataset* se entrena el modelo con los hiperparámetros seleccionados y obtienen las métricas de evaluación Macro-average_all y se compara con el resultado sin balanceo, como se ve en la Tabla 28.

La aplicación de sobremuestreo produjo un incremento el *Recall*, acompañado de una ligera disminución en *precision*, *Accuracy* y $F1_score$. Este comportamiento sugiere que el balanceo favoreció una mayor detección de clases minoritarias a costa de un aumento en falsos positivos.

Aunque el sobremuestreo mejoró un poco el *Recall*, la disminución en las otras métricas Macro-average_all, es motivo para evitar el balanceo para el modelo final.

6.5.4 Resultado de los experimentos

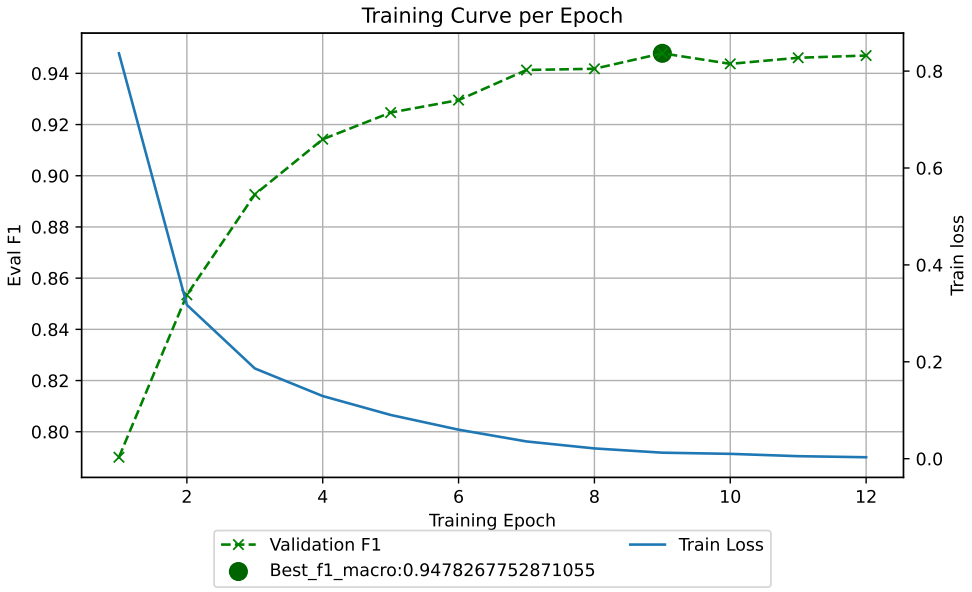
Gracias a los experimentos, se seleccionan los hiperparámetros y la configuración final para el proceso de adaptación y evaluación del modelo, obteniendo el modelo *MODEL_re* con un tiempo de entrenamiento de 10 minutos y 52 segundos:

- *GPU = NVIDIA RTX 4500 Ada Generation de 24 GB*
- *Longitud máxima de secuencia (max_length) = 512*
- *Número máximo de Epochs = 12*
- $Freeze_Layers = 0$
- *Sin Balanceo de clases*
- *epoch* máximo 12
- $dropout = 0.1$
- *Per-device batch size* $= 4$
- *Gradient accumulation steps* $= 4$
- *Effective batch size* (*Per-device batch size* $\times$ *Gradient accumulation steps*) $= 16$
- *Paciencia (Early_stopping)*$= 3$
- *Límite(Early_stopping)* $= 2e-03$

- *Learning rate* $= 5e-05$
- *Weight decay* $= 0.01$
- *Semilla aleatoria* $= 42$

Figura 29

Gráfica de las curvas de entrenamiento del modelo RE (MODEL_re) seleccionado.



Nota. Se muestran las curvas F1 vs época (Epoch) y pérdida (Loss) vs época (Epoch).

6.5.5 Evaluación del modelo sobre el conjunto de validación

El modelo *MODEL_re* es evaluado con 1180 pares entidades-relación con 132 oraciones únicas del conjunto de validación. Se obtiene las métricas de evaluación guardadas en el EXCEL *Metricas_re.xlsx*. En la Tabla 29 se ve un resumen de las métricas de evaluación.

Tabla 29

Métricas de evaluación del modelo MODEL_re sobre el conjunto de validación.

Métrica	Micro-average	Macro-average_all	Macro-average_rel	Métrica	Valor
Precision	0.9661	0.9488	0.8	**Accuracy**	0.9661
Recall	0.9661	0.9472	0.7522		
F1-score	0.9661	0.9478	0.7747		

Métricas por clase

Clase	Precision	Recall	F1-score	support
HAS_VALUE	0.9286	0.9512	0.9398	41
NO_RELATION	0.9751	0.9751	0.9751	402
PERFORMED_UNDER_CONDITION	0.9796	0.96	0.9697	50
PREPARED_WITH	0.8947	0.85	0.8718	40
USED_WITH	0.9661	1	0.9828	57
accuracy			0.9661	590
Macro-average_all avg	0.9488	0.9473	0.9478	590
weighted_all avg	0.9659	0.9661	0.9659	590

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar, y el macro-average_rel calcula la métrica por clase, sin incluir la clase "NO_RELATON" y luego promedia sin ponderar.

El valor superior de la métrica *micro-average* respecto a *macro-average_all* sugiere un mejor desempeño sobre las clases con mayor representación dentro del conjunto de validación. También, esta afirmación se reafirma al ver la diferencia entre las métricas *macro-average_all* y *Macro-average_Rel*, donde se muestra el peso de la clase negativa "*NO_RELATION*", una clase mayoritaria, dentro del conjunto de evaluación. Estos resultados corresponden a la evaluación empleada durante el ajuste y selección del modelo.

6.5.6 Validación cruzada del modelo

Tabla 30

Métricas promedio obtenidas en validación cruzada por k-Folds estratificados del modelo MODEL_re, con $K = 10$.

Métrica		**Micro-average**	**Macro-average_all**	**Macro-average_rel**
Precision	**mean**	0.9652	0.9461	0.7995
	std	0.0088	0.0144	0.0015
Recall	**mean**	0.9652	0.9582	0.7643
	std	0.0088	0.0178	0.0184
F1-score	**mean**	0.9652	0.9512	0.7812
	std	0.0088	0.0123	0.0096

Métrica		**Valor**
Accuracy	**mean**	0.9652
	std	0.0088

Métricas por clase

Clase		**Precision**	**Recall**	**F1-score**	**support**
HAS_VALUE	**mean**	0.9660	0.9679	0.9666	41
	std	0.0251	0.0311	0.0185	
NO_RELATION	**mean**	0.9795	0.9696	0.9745	402
	std	0.0108	0.0115	0.0065	
PERFORMED_UNDER _CONDITION	**mean**	0.9721	0.9601	0.9659	50
	std	0.0250	0.0211	0.0185	
PREPARED_WITH	**mean**	0.8815	0.9355	0.9058	40
	std	0.0585	0.0499	0.0361	
USED_WITH	**mean**	0.9312	0.9579	0.9432	57
	std	0.0450	0.0381	0.0262	

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar, y el macro-average_rel calcula la métrica por clase, sin incluir la clase "NO_RELATON" y luego promedia sin ponderar.

La validación cruzada por K-Fold estratificado del modelo *MODEL_re* se hace con $k = 10$. Los resultados se guardan en el EXCEL $MODEL_re_10k_Kfold_metricas.xlsx$. En la Tabla 30 se ve un resumen de las métricas obtenidas al validar el modelo.

Los resultados obtenidos mediante validación cruzada por k-Fold estratificado fueron similares a los observados en la evaluación con el conjunto de validación, presentando desviaciones estándar reducidas. Esto sugiere que funciono el proceso para optimizar los hiperparámetros, generando un entrenamiento estable y una baja sensibilidad frente a la partición específica de los datos.

6.5.7 Evaluación (Test) independiente del modelo

El modelo *MODEL_re* es evaluado con los 3934 pares entidades-relación con 562 oraciones únicas del conjunto de evaluación independiente. Se obtiene las métricas de desempeño guardadas en el EXCEL *ScoreModel_re_Evaluation.xlsx*. En la Tabla 31 se ve un resumen de las métricas de evaluación.

En general, el comportamiento observado durante la evaluación con el conjunto de validación (Tabla 29), la validación cruzada por K-folds (Tabla 30) y el *test* independiente, presentan un comportamiento equivalente, donde el valor de la métrica *micro-average* es superior respecto al *macro-average_all*, sin embargo, con un valor disminuido de las métricas en el *test* independiente, como se ve en la Tabla 31.

Tabla 31

Desempeño del modelo MODEL_re sobre un Test independiente.

Métrica	Micro-average	Macro-average_all	Macro-average_rel	Métrica	Valor
Precision	0.9174	0.8896	0.8000	**Accuracy**	0.9174
Recall	0.9174	0.9064	0.7218		
F1-score	0.9174	0.8942	0.7579		

Métricas por clase

Clase	Precision	Recall	F1-score	support
HAS_VALUE	0.7556	0.9611	0.8461	386
NO_RELATION	0.9486	0.9230	0.9356	2559
PERFORMED_UNDER_CONDITION	0.9591	0.8523	0.9025	440
PREPARED_WITH	0.8955	0.8257	0.8592	218
USED_WITH	0.8892	0.9698	0.9277	331
accuracy			0.9174	3934
Macro-average_all avg	0.8896	0.9064	0.8942	3934
weighted_all avg	0.9229	0.9174	0.9182	3934

Nota. El micro-average calcula las métricas globalmente considerando todas las predicciones. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar, y el macro-average_rel calcula la métrica por clase, sin incluir la clase "NO_RELATON" y luego promedia sin ponderar.

En el *Test* independiente la principal diferencia se observa en la clase "*HAS_VALUE*", donde la Precisión muestra la mayor disminución de su valor. Esta clase tiene una de las menores representaciones entre los ejemplos del conjunto utilizado para el entrenamiento del modelo, lo que puede limitar la variedad de patrones aprendidos. Las diferencias en la distribución y en las formas de expresión presentes en el conjunto de *test* también podría haber afectado su capacidad de generalización.

Otro posible factor puede estar relacionado con la dependencia entre las etapas de extracción de entidades (NER) y extracción de relaciones (RE). Esta clase incluye la entidad ("*NFF*") que presentó uno de los menores desempeños entre las categorías evaluadas en el modelo NER, así, los errores pudieron haberse propagado desde la etapa NER hasta la RE, afectando aquellas relaciones que dependen de su correcta identificación.

Adicionalmente, el alto valor de las métricas obtenido para la clase "*NO_RELATION*" se observa, considerando el procedimiento utilizado para construir los ejemplos negativos. Dado que esta clase incorpora parcialmente un componente determinista derivado del esquema de relaciones definido, incorporado deliberadamente para proporcionar al modelo ejemplos de pares sin relación. Por eso su $F1 - score$ no representa una medida de validación humana equivalente a la utilizada para las demás clases. A raíz de esto, el desempeño global del modelo se analiza conjuntamente con las métricas por clase y con el comportamiento observado en el conjunto de *test* independiente.

6.6 Construcción automática de la Base de datos (Database) a partir de la fase de inferencia

Con los modelos *MODEL_ner* y *MODEL_re* desarrollados, se realiza la fase de inferencia sobre el sistema de archivos con 1309 artículos HTML y PDF de la carpeta *DATA_TRAIN* y se obtienen dos carpetas, *DATA_NER* y *DATA_RE* con los resultados respectivos de cada modelo, tal como se describe en la Sección 5.8. Usando el ordenador personal descrito en la Sección 5.11, fueron necesarios 28 minutos con 45 segundos para obtener la carpeta *DATA_NER* con archivos JSON producto de la fase de inferencia del modelo NER sobre la carpeta *DATA_TRAIN*. Durante el proceso no fue posible extraer el texto de 2 artículos en PDF (*PDFs*_148.*pdf* y *PDFs*_856.*pdf*) dejando resultados de 1307 artículos en archivos JSON y un archivo TXT de registro.

Uno de los archivos de la carpeta *DATA_NER* es *DATANER.json* un diccionario de diccionarios con todas las entidades encontradas en cada artículo. Los demás corresponden cada uno a un artículo de la carpeta *DATA_TRAIN* y contienen la información de secciones del texto, en varias ventanas de 2 oraciones junto a sus entidades, como se ve en la estructura de datos descrita en la Sub-sección 5.8.1. En el archivo TXT de registro *Infodata_NER_DATA.txt* se encuentra la distribución de cada artículo, las secciones extraídas y el recuento de caracteres. Los nombres Normalizados de las secciones guardadas y un ejemplo de como esta escrito en el TXT la distribución uno de los artículos se ven a continuación.

Code Listing 6.3. Nombres normalizados de las secciones que se extraen de los artículos.

```
[
        "Abstract",
        "Summary",
        "Context and Scale",
        "Executive Summary",
        "The Bigger Picture",
        "Methods",
        "Materials and Methods",
        "Methodology",
        "Experimental",
        "Data",
        "Materials and Thin Film Preparation",
        "Results",
        "Results and Discussion",
        "Analysis",
        "Discussion",
        "Concluding remarks",
        "Future Work",
        "Conclusions",
        "Texto completo"
]
```

Code Listing 6.4. Extracto del archivo TXT de registro.

```
------------------------------------------------
art{\'i}culo N.0 DE 199
Extrayedo las entidades HTMLs_10.html
El art{\'i}culo tiene las secciones que nos interesan dict_keys(["
   Abstract", "Methods", "Results", "Conclusions"])
Con un total 11730 caracteres
Archivo ./DATA_NER/NER_htmls_10.json guardado
Tiempo para un art{\'i}culo en HTML: 0:00:00.820494
------------------------------------------------
```

Para obtener la carpeta *DATA_RE* con 1307 archivos JSON producto de la fase de inferencia del modelo RE sobre la carpeta *DATA_NER* fueron necesarios 11 horas y 56 minutos usando la Workstation descrita en la Sección 5.11. Cada archivo JSON contiene la información de secciones del texto, en varias ventanas de 2 oraciones con pares entidades-relación, como se ve en la estructura de datos descrita en la Sub-sección 5.8.2.

6.7 Post-procesamiento: la Reconstrucción de devices

Como se describió en la Sección 5.9 se desarrollo una serie de Scripts para hacer lo que se decidio denominar *extracción de relaciones a nivel de dispositivo/device*, un acercamiento a la extracción de relaciones a nivel de documento. Los devices son agrupaciones de entidades a nivel de sección, agrupados alrededor de uno de los tres tipos de clases de entidad (capas, desempeño o métodos), validados por las relaciones inferidas por el modelo *MODEL_re* y guardadas en los 1307 archivos JSON de la carpeta *DATA_RE*.

Para generar *Database* final: los 1307 archivos JSON almacenados en la carpeta *DATA_RESULTS*, fueron necesarios 44 minutos y 36 segundos usando el ordenador personal descrito en la Sección 5.11.

El archivo TXT de registro *Infodata_SEC_DATA_RESULTS.txt* describe el numero de devices reconstruidos en cada sección y alrededor de que clase de entidad se reconstruyo. Un ejemplo de como esta escrito en el TXT la distribución de los devices en uno de los artículos se ve a continuación.

Code Listing 6.5. Extracto del archivo TXT de registro.

```
------------------------------------------------
art{\'i}culo N.424 DE 499

Agrupando las relaciones del art{\'i}culo ./RE_NER_htmls_620
SECCION Abstract:
Se agruparon los devices alrededor del performance. Se agruparon 2
   devices
SECCION Texto completo:
Se agruparon los devices alrededor del performance. Se agruparon 3
   devices
Se agruparon los devices alrededor de los methods. Se agruparon 3
   devices

Se obtuvo la informaci{\'o}n de las secciones del art{\'i}culo :
   dict_keys(["Abstract", "Texto completo"])

Archivo ./DATA_RESULTS/DATA_RE_NER_htmls_620.json guardado

Tiempo para la extraccion las relaciones entre las entidades a nivel
    de seccion del art{\'i}culo RE_NER_htmls_620: 0:00:04.833271
------------------------------------------------
```

6.8 Validación de los Resultados

Para validar los resultados del proceso de inferencia y post-procesamiento como se describió en la Sección 5.10, se realizan dos comparaciones vs los datos descargados desde el *Database* manual (Jacobsson et al., 2022), un subconjunto con 29070 registros correspondientes a 5107 artículos sobre celdas solares de perovskita convencionales, publicados entre los años 2003 y 2020: la extracción de entidades de cada artículo y los devices reconstruidos.

6.8.1 Comparación a nivel de entidad

En la carpeta *Comparison_ner* se almacenan los archivos JSON con los resultados de las tres métricas de comparación entre las entidades extraídas (*DATANER.json*) y el *Database* manual: *Precision*, *Recall* y *Containment* (coeficiente de solapamiento, *overlap coefficient*). En la Tabla 32 se ve un resumen de los resultados de las métricas y la Tabla 33 el valor promedio de las métricas.

Tabla 32

Resumen de la comparación y validación a nivel de entidades.

Métrica	**ETL**	**PTF**	**HTL**	**VOC**
Coeficiente de solapamiento	0.8930	0.6731	0.9377	0.7733
precisión	0.5445	0.2085	0.3483	0.4775
Recall	0.8672	0.6712	0.8920	0.3283
Entidades del Dataset manual	1765	1610	1470	5996
Entidades correctamente predichas	1567	1001	1328	1613
Métrica	**JSC**	**FF**	**PCE**	**MTDS**
Coeficiente de solapamiento	0.7861	1	0.6881	0.8863
precisión	0.4758	0.7713	0.4447	0.0975
Recall	0.3191	0.3547	0.4960	0.8771
Entidades del Dataset manual	6910	6385	7131	2017
Entidades correctamente predichas	1865	2041	3268	1683

Tabla 33

Valores promedio de la comparación y validación a nivel de entidades.

Métrica	**Valor**
Coeficiente de solapamiento	0.8537
precisión	0.4221
Recall	0.6021
Entidades del Dataset manual	33284
Entidades correctamente predichas	14366
Artículos	1307

Durante el análisis de errores para comprobar el porque del bajo porcentaje de acierto (*Percentage of success*) en las entidades VOC, PCE, JSC y FF respecto a las entidades consignadas en el *Database* manual, se observa que algunas de estas entidades no se encontraban explícitamente en el texto principal de ciertos artículos procesados, sino en tablas o documentos suplementarios. Debido a que el *Pipeline* fue diseñado para operar únicamente sobre el texto del artículo, dichas entidades no podían ser recuperadas automáticamente, lo que afectó las métricas de extracción correspondientes. Aunque la extracción de información desde tablas fue abordada durante el desarrollo del trabajo, esta funcionalidad no se incorporó al *Pipeline* principal debido a que no se encuentra completamente desarrollada, ver Apéndice D.

6.8.2 Comparación a nivel de devices

En la carpeta *Comparison* se almacenan los archivos JSON con los resultados de las tres métricas de comparación entre los devices reconstruidos (*DATA_RESULTS*) y el *Database* manual: *Precision*, *Recall* y *Containment* (coeficiente de solapamiento, *overlap coefficient*). En la Tabla 34 se ve un promedio del valor de las métricas.

Tabla 34

Valores promedio de la comparación y validación a nivel de devices.

Métrica	Valor
Coeficiente de solapamiento	0.8733
Coeficiente de solapamiento de las estructuras	0.5636
precisión	0.5099
precisión de las estructuras	0.5062
Recall	0.0890
Recall de las estructuras	0.2216
Artículos	1307

El modelo RE y el proceso de post-procesamiento usados para generar los archivos de la carpeta *DATA_RESULTS* con la lista de devices reconstruidos, presentan un comportamiento conservador, que prioriza las predicciones correctas a costa de omitir parte de las relaciones reales presentes en el texto. Esto provoca la pérdida de algunos *devices* o de entidades que deberían formar parte de ellos. Este comportamiento podría verse aumentado por la ausencia de algunas entidades dentro del documento principal de ciertos artículos, como se mostró en la sub-sección anterior, la comparación a nivel de entidad.

6.8.3 Análisis de un artículo

Con el objetivo de realizar un análisis de errores complementario a la validación cuantitativa presentada en la sección anterior, se examina de forma individual un artículo representativo del conjunto de datos. Se Analiza el artículo *HTMLs*_620.*html* llamado "*Conversion efficiency improvement of inverted CH3NH3PbI3 perovskite solar cells with room temperature sputtered ZnO by adding the C60 interlayer*" almacenado en el sistema de archivos de la carpeta *DATA_TRAIN*. En la Tabla 35 se ven los resultados de las entidades extraídas vs el *Database* manual, en la Tabla 36 se presenta la información extraída para un dispositivo fotovoltaico vs el *Database* manual y en la Tabla 38 una reconstrucción de la Figura 33 donde se observa una tabla del documento que se logro extraer con el proceso del Apéndice D.

La Tabla 37 presenta las métricas de coherencia calculadas para agrupar el dispositivo. La cobertura de relaciones indica la proporción de relaciones verificadas respecto a las inferidas por el modelo, mientras que el soporte posicional evalúa si las entidades se encuentran asociadas al

dispositivo correcto dentro del texto. Valores cercanos a 1 representan una mayor consistencia de la extracción.

Tabla 35

Comparación de las entidades extraídas del artículo HTMLs_*620.html vs el Database manual.*

Etiqueta	Extracción automática	Database manual
ETL	ZnO, C60, BCP, ETL, TiO2	C60, ZnO-c, BCP
HTL	PEDOT:PSS	PEDOT:PSS
PTF	MAPbI3	MAPbI3
VOC	0.91, 0.9	0.25, 0.9, 0.91
PCE	10.93, 10.6	1.32, 9.07, 10.6, 10.93
JSC	16.23, 19.41	18.45, 18.11, 16.23, 19.41
FF	-	0.289, 0.555, 0.7240, 0.62
MTDS	evaporation, sputtering, spin-coating, spin-coated	Evaporation, Sputtering, Spin-coating

Tabla 36

Comparación de un device reconstruido del artículo HTMLs_*620.html vs los valores encontrados en el Database manual.*

Campo	**Extracción automática**
Dispositivo ID	0
Estructura	ITO | PEDOT:PSS | perovskite | C60 | RT sputtered ZnO | Al
Capas identificadas	C60, ZnO, $CH_3NH_3PbI_3$, PEDOT:PSS
Métodos identificados	Sputtering, room temperature
JSC	19.41 mA
VOC	0.91 V
PCE	10.93 %
FF	–
Campo	**Database manual**
Estructura	SLG | ITO | PEDOT:PSS | Perovskite | C60 | ZnO-c | Al
Capas	C60 | ZnO-c, $MAPbI_3$, PEDOT:PSS,
Métodos	Spin-coating, Evaporation, Sputtering,
JSC	19.41 mA
VOC	0.91 V
PCE	10.93 %
FF	0.62

Tabla 37

Métricas de coherencia calculadas para el dispositivo de la Tabla 36.

Métrica	Optimista	Realista
Relaciones verificadas	14	40
Relaciones inferidas	14	53
Cobertura de relaciones	1.000	0.755
Soporte de entidades	1.000	1.000
Soporte posicional	–	0.907
Confianza final	1.000	0.859

Nota. La métrica Optimista considera únicamente la existencia de la relación. La métrica Realista exige además que dicha relación esté correctamente asociada al *device* correspondiente según su ubicación en el texto. Ambas métricas se definen en la Figura 25 y en el Apéndice F.

Tabla 38

Tabla extraída del artículo HTMLs_*620.html*.

Devices	JSC (mA/cm2)	VOC (V)	FF (%)	η (%)
C60/BCP	16.23	0.90	72.4	10.60
ZnO	-	-	-	-
5 nm-C60/ZnO	18.45	0.25	28.9	1.32
15 nm-C60/ZnO	19.41	0.91	62.0	10.93
30 nm-C60/ZnO	18.11	0.90	55.5	9.07

Nota. Reproducción de la tabla de la Figura 33 contenida en un archivo EXCEL, generado mediante el proceso de extracción de tablas descrito en el Apéndice D, aplicado al artículo *HTMLs_*620*.html*.

Para el artículo analizado, la métrica optimista alcanzó un valor de 1.0, indicando que todas las entidades extraídas encontraron soporte dentro del conjunto global de relaciones inferidas. Por otra parte, la métrica realista obtuvo una confianza de 0.859, evidenciando que la mayoría de

las entidades no solo fueron identificadas correctamente, sino también asociadas al dispositivo correspondiente dentro del texto.

El ejemplo presentado constituye un caso representativo de las limitaciones discutidas previamente al observar los resultados de las comparaciones. En particular, se observa que algunas entidades relacionadas con el desempeño de la celda no son extraídas debido a que la información se encuentra contenida en tablas del artículo, las cuales no forman parte del texto utilizado durante la inferencia.

6.9 Limitaciones

- Como se detalla en el Apéndice D, debido a la complejidad estructural de las tablas presentes en los artículos científicos, la información contenida en estos elementos no fue incorporada al proceso de agrupación de devices durante el post-procesamiento ni a las etapas de validación mediante comparaciones. En consecuencia, los resultados obtenidos se limitan a la información extraída del texto principal de los documentos.

- El *pipeline* no procesa los documentos suplementarios asociados a los artículos científicos, por lo que una fracción de la información experimental reportada por los autores puede no estar representada en la base de datos generada.

- La heterogeneidad en las estructuras PDF y HTML, dependientes del Journal/editorial o plataforma de publicación, obliga a revisar y adaptar los Scripts de descarga y segmentación cada vez que se desee ampliar el corpus, para asegurar que las nuevas estructuras estén contempladas.

- Algunas relaciones semánticas entre entidades requieren información distribuida entre múltiples oraciones o incluso en diferentes secciones. Dado que el modelo emplea principalmente contexto local, este tipo de relaciones puede no ser identificado de forma consistente.

- La calidad de los modelos entrenados depende directamente de la consistencia y cobertura de los datos etiquetados. Posibles sesgos u omisiones humanas, presentes durante el proceso de anotación pueden propagarse a las etapas posteriores y afectar los resultados.

- La evaluación se restringió a los modelos y configuraciones descritos en este trabajo. La exploración de arquitecturas adicionales, estrategias de entrenamiento alternativas y experimentos de mayor escala queda planteada como trabajo futuro.

6.10 Disponibilidad de datos

Los archivos resultantes generados durante esta investigación, incluyendo los modelos entrenados y estructuras JSON y EXCEL derivados del análisis, se encuentran disponibles en un repositorio de GitHub y una carpeta de Drive. Por favor solicitar el acceso mediante correo electrónico al autor: `mary2238319@correo.uis.edu.co`.

Debido al tamaño de los archivos generados, estos recursos no se adjuntan directamente al documento, sino que se almacenan externamente para facilitar su consulta y reproducción. Las Tablas del Apéndice I relacionan cada módulo implementado con la sección metodológica y carpeta correspondiente.

7 Conclusiones

A partir del proceso presentado y los resultados obtenidos en el presente trabajo de grado, es posible obtener la siguientes conclusiones.

- Se logra desarrollar los algoritmos de recolección de datos de la web y la generación semi-automática del conjunto de datos etiquetados, reduciendo la necesidad de intervención humana en el entrenamiento de modelos.

- Se desarrolla la extracción a nivel de secciones de texto mediante dispositivos/devices que se denomino *extracción de relaciones a nivel de dispositivo/device*, para el cual se entrenan dos modelos *Deep Learning*, un modelo para la tarea de identificación de entidades nombradas (NER) y otro para la tarea de extracción de relaciones (RE) a nivel de oración, conectados en secuencia en un *Pipeline*; y se hace uso de técnicas basadas en la Ingeniería del Conocimiento (*Know-ledge Engineering*, KE) para escalar el nivel la extracción de relaciones.

- Se obtiene un buen desempeño de los modelos NER y RE sobre el conjunto de validación, como se ve en la Tabla 39. Asimismo, la evaluación sobre el conjunto de evaluación (*Test*) independiente permite observar el desempeño de los modelos frente a datos que no participaron en su entrenamiento ni en el proceso de selección de hiperparámetros. En ambos modelos se observa una disminución de las métricas respecto al conjunto de validación, manteniéndose un desempeño satisfactorio sobre el conjunto de *Test* independiente.

Tabla 39

Resumen del desempeño de los modelos NER y RE sobre los conjuntos de validación y Test independiente.

		Micro			Macro_all		
Modelo	**Conjunto**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
NER	Validación	0.95	0.95	0.95	0.93	0.93	0.93
	Test independiente	0.91	0.90	0.91	0.88	0.88	0.87
RE	Validación	0.97	0.97	0.97	0.95	0.95	0.95
	Test independiente	0.91	0.91	0.91	0.89	0.91	0.89

Además, en los resultados del desempeño global del modelo RE se debe tener en cuenta que las métricas correspondientes a la clase "*NO_RELATION*" deben interpretarse considerando su componente de generación determinista presente en su proceso de construcción y etiquetado. Esta consideración es relevante en los conjuntos de datos (*Dataset*) de entrenamiento, validación y el conjunto de evaluación (*Test*).

- Los resultados finales de la reconstrucción automática de dispositivos, obtenidos mediante el modelo NER, RE y el proceso de post-procesamiento basado en reglas de conocimiento, mostraron un desempeño condicionado por las limitaciones de cobertura de la información disponible en los documentos y por el comportamiento conservador del proceso de inferencia. En esta evaluación se obtuvieron valores promedio de $Containment_score = 0.87$, $Precision_score = 0.51$ y $Recall_score = 0.09$. Estos resultados se analizaron en función de la disponibilidad de las entidades en los documentos principales, la distribución de la información entre diferentes secciones y las restricciones de contexto empleadas para la reconstrucción de los dispositivos.

- Obviando las limitaciones descritas, el *Pipeline* planteado y usado en este trabajo, tiene el potencial de ser aplicado a diferentes dominós científicos e integrarle más implementos, como avanzar en la extracción de tablas o adicionar la información de archivos suplementarios.

- La metodología propuesta demostró que es posible desarrollar sistemas de reconocimiento de entidades (NER), extracción de relaciones (RE) y construcción automática de bases de datos científicas dentro de un único *pipeline* basado en herramientas de código abierto. Esta integración permitió transformar información no estructurada proveniente de artículos científicos en una representación estructurada susceptible de análisis y reutilización posterior.

- Se logró un equilibrio favorable entre desempeño y costo computacional mediante el uso de modelos de lenguaje preentrenados (PLMs) de tamaño intermedio y estrategias de post-procesamiento. Los resultados obtenidos demuestran que es posible desarrollar un *pipeline* de extracción de información científica con recursos computacionales moderados. El uso de una *workstation* con GPU fue necesario únicamente en etapas específicas del entrenamiento y evaluación del modelo de extracción de relaciones (RE), mientras que los demás componentes del sistema pudieron ejecutarse en un ordenador personal convencional.

Referencias Bibliográficas

Abdullah, M. H. A., Aziz, N., Abdulkadir, S. J., Alhussian, H. S. A., and Talpur, N. (2023). Systematic literature review of information extraction from textual data: recent methods, applications, trends, and challenges. *IEEE Access*, 11:10535–10562.

Alias, G. and Cassanelli, R. (2019). *NLP aplicado a análisis de texto*. PhD thesis, Universidad Nacional de Mar del Plata. Facultad de Ingeniería. Argentina.

Alyafeai, Z., Alshaibani, M. S., and Ahmad, I. (2020). A survey on transfer learning in natural language processing. *ArXiv*, abs/2007.04239.

Alzubaidi, L., Bai, J., Al-Sabaawi, A., Santamaría, J., Albahri, A. S., Al-dabbagh, B. S. N., Fadhel, M. A., Manoufali, M., Zhang, J., Al-Timemy, A. H., et al. (2023). A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1):46.

Amini, M.-R., Feofanov, V., Pauletto, L., Hadjadj, L., Devijver, E., and Maximov, Y. (2024). Self-training: A survey.

Antona Castañares, A. (2020). *Aplicación del dropout a la cuantificación de la incertidumbre en redes neuronales*. PhD thesis, Industriales.

Ariga, A. (2021). tabula,`https://pypi.org/project/tabula-py/`.

AthenaTech LLC (2019). Ai, machine learning (ml) and natural language processing (nlp), https://athenatech.tech/f/ai-machine-learning-ml-and-natural-language-processing-nlp.

Baciero Fernández, J. I. (2020). Elaboración de un modelo de reconocimiento de entidades nominales (ner) para su uso en aplicaciones de procesamiento del lenguaje natural (nlp).

Beard, E. J. and Cole, J. M. (2022). Perovskite-and dye-sensitized solar-cell device databases auto-generated using chemdataextractor. *Scientific Data*, 9(1):329.

Beltagy, I., Lo, K., and Cohan, A. (2019). Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y., Goodfellow, I., Courville, A., et al. (2017). *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Bharadiya, J. (2023). A comprehensive survey of deep learning techniques natural language processing. *European Journal of Technology*, 7(1):58–66.

Bismart (2021). ¿cuál es la diferencia entre el machine learning y el deep learning? , https://blog.bismart.com/es/diferencia-machine-learning-deep-learning.

C.B.Honsberg and S.G.Bowden (2019). Photovoltaics education website, `https://www.pveducation.org/`.

Chakkarwar, V., Tamane, S., and Thombre, A. (2023). A review on bert and its implementation in various nlp tasks. In *International Conference on Applications of Machine Intelligence and Data Analytics (ICAMIDA 2022)*, pages 112–121. Atlantis Press.

Chauhan, N. K. and Singh, K. (2018). A review on conventional machine learning vs deep learning. In *2018 International conference on computing, power and communication technologies (GUCON)*, pages 347–352. IEEE.

Chen, M., Yin, Z., Shan, Z., Zheng, X., Liu, L., Dai, Z., Zhang, J., Liu, S. F., and Xu, Z. (2024). Application of machine learning in perovskite materials and devices: A review. *Journal of Energy Chemistry*, 94:254–272.

Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., Persson, K. A., and Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Danenas, P. and Skersys, T. (2022). Exploring natural language processing in model-to-model transformations. *IEEE Access*, 10:116942–116958.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Diaz-Garcia, J. A. and Lopez, J. A. D. (2025). A survey on cutting-edge relation extraction techniques based on language models. *Artificial Intelligence Review*, 58(9):287.

Eisenstein, J. (2019). *Introduction to natural language processing*. MIT press.

El Asikri, M., Knit, S., and Chaib, H. (2020). Using web scraping in a knowledge environment to build ontologies using python and scrapy. *European Journal of Molecular & Clinical Medicine*, 7(03):2020.

Elangovan, N. K., Kannadasan, R., Beenarani, B., Alsharif, M. H., Kim, M.-K., and Hasan Inamul, Z. (2024). Recent developments in perovskite materials, fabrication techniques, band gap engineering, and the stability of perovskite solar cells. *Energy Reports*, 11:1171–1190.

Essien, I. A., Adebayo, A. O., Afuwape, A. A., Eboseremen, B. O., Oladega, F., and Soneye, O. M. (2023). The ethics of web scraping in research: A review: Investigating the boundaries, legal implications, and societal acceptance of web scraping as a data collection method. *Journal of Frontiers in Multidisciplinary Research*, 4(1):529–538.

Face, H. (2026a). Hugging face model hub. `https://huggingface.co/models`. Repositorio oficial de modelos preentrenados.

Face, H. (2026b). Transformers: State-of-the-art machine learning for nlp, vision, audio, and multimodal models. `https://huggingface.co/docs/transformers`.

Fan, L., Li, L., Ma, Z., Lee, S., Yu, H., and Hemphill, L. (2023). A bibliometric review of large language models research from 2017 to 2023. *arXiv preprint arXiv:2304.02020*.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. S. (2017). Allennlp: A deep semantic natural language processing platform.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Springer Cham.

Golshan, P. N., Dashti, H. R., Azizi, S., and Safari, L. (2018). A study of recent contributions on information extraction. *arXiv preprint arXiv:1803.05667*.

Google Cloud (2021). Vertex ai, `https://cloud.google.com/vertex-ai#section-5`.

Göpfert, J., Kuckertz, P., Müller, G., Lütz, L., Körner, C., Khuat, H., Stolten, D., and Weinand, J. M. (2026). Quinex: Quantitative information extraction from text using open and lightweight llms. *The Innovation*.

Gupta, T., Zaki, M., Krishnan, N. A., and Mausam (2022). Matscibert: A materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1):102.

Hawizy, L., Jessop, D. M., Adams, N., and Murray-Rust, P. (2011). Chemicaltagger: A tool for semantic text-mining in chemistry. *Journal of cheminformatics*, 3:1–13.

Heidenreich, H. (2018). Stemming? Lemmatization? What?, `https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8`.

Herrmann, M. (2021). Qt designer download, https://build-system.fman.io/qt-designer-download.

Himanen, L., Geurts, A., Foster, A. S., and Rinke, P. (2019). Data-driven materials science: Status, challenges, and perspectives. *Advanced Science*, 6(21):1900808.

Hobbs, J. R. and Riloff, E. (2010). Information extraction. *Handbook of natural language processing*, 15:16.

Hong, Z. (2023). Chatgpt for computational materials science: A perspective. *Energy Material Advances*, 4:0026.

Honnibal, M., Montani, I., et al. (2015). Spacy: industrial-strength natural language processing (nlp) with python and cython, https://spacy.io.

Honnibal, M., Montani, I., et al. (2026a). Spacy:api, https://spacy.io/api.

Honnibal, M., Montani, I., et al. (2026b). Spacy:usage, https://spacy.io/usage/spacy-101.

Hu, D., Liu, B., Zhu, X., Lu, X., and Wu, N. (2024a). Zero-shot information extraction from radiological reports using chatgpt. *International Journal of Medical Informatics*, 183:105321.

Hu, Z., Hou, W., and Liu, X. (2024b). Deep learning for named entity recognition: a survey. *Neural Computing and Applications*, 36(16):8995–9022.

Huang, S. and Cole, J. M. (2022). Batterydataextractor: battery-aware text-mining software embedded with bert models. *Chem. Sci.*, 13:11487–11495.

IBM (2014). Valores de archivos pdf, https://www.ibm.com/docs/es/cognos-analytics/10.2.2?topic=SSEP7J_10.2.2/com.ibm.swg.ba.cognos.ug_cra.10.2.2.doc/c_pdffilesettings.html.

Jacobsson, T. J., Unger, E., et al. (2022). An open-access database and analysis tool for perovskite solar cells based on the fair data principles. *Nature Energy*, 7(1):107–115.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning: with applications in R*, volume 103. Springer.

Jeon, N. J., Lee, J., Noh, J. H., Nazeeruddin, M. K., GraÌĹtzel, M., and Seok, S. I. (2013). Efficient inorganic–organic hybrid perovskite solar cells based on pyrene arylamine derivatives as hole-transporting materials. *Journal of the American Chemical Society*, 135(51):19087–19090.

Jeong, J., Kim, M., Seo, J., Lu, H., Ahlawat, P., Mishra, A., Yang, Y., Hope, M. A., Eickemeyer, F. T., Kim, M., et al. (2021). Pseudo-halide anion engineering for α-fapbi3 perovskite solar cells. *Nature*, 592(7854):381–385.

Jessop, D. M., Adams, S. E., Willighagen, E. L., Hawizy, L., and Murray-Rust, P. (2011). Oscar4: a flexible architecture for chemical text-mining. *Journal of cheminformatics*, 3(1):1–12.

Jiménez-Díaz, M., Camacho-Parra, C., Botero-Londoño, M., Sepúlveda-Sepúlveda, A., and Otálora-Bastidas, C. (2022). *Herramienta de software para la extracción automática de parámetros de desempeño a partir de publicaciones científicas de celdas solares de perovski-*

ta. Tesis de pregrado, Universidad Industrial de Santander, Bucaramanga, Colombia. https://noesis.uis.edu.co/handle/20.500.14071/9977.

Jiménez-Díaz, M. Z., Sepúlveda-Sepúlveda, A., and Botero-Londoño, M. (2025). Named entity recognition for performance and synthesis information of perovskite solar cells using spacy. In *International Conference on Computational Science and Its Applications*, pages 303–319. Springer.

Jiménez Ross, M. A. (2017). Fabricacion y caracterización de celdas solares orgánicas basadas en perovskita. Tesis de ingeniería química, Universidad de Sonora, Mexico.

Jurafsky, D. and Martin, J. H. (2026). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition. Online manuscript released January 6, 2026.

Kadulkar, S., Sherman, Z. M., Ganesan, V., and Truskett, T. M. (2022). Machine learning-assisted design of material properties. *Annual Review of Chemical and Biomolecular Engineering*, 13(1):235–254.

Kamath, U., Liu, J., and Whitaker, J. (2019). *Deep learning for NLP and speech recognition*, volume 84. Springer.

Kang, Y., Cai, Z., Tan, C.-W., Huang, Q., and Liu, H. (2020). Natural language processing (nlp) in management research: A literature review. *Journal of Management Analytics*, 7(2):139–172.

Kenneth Leung (2022). Micro, macro weighted averages of f1 score, clearly explained, https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f/.

Kojima, A., Teshima, K., Shirai, Y., and Miyasaka, T. (2009). Organometal halide perovskites as visible-light sensitizers for photovoltaic cells. *Journal of the American Chemical Society*, 131(17):6050–6051.

Krallinger, M., Rabal, O., Lourenço, A., Oyarzabal, J., and Valencia, A. (2017). Information retrieval and text mining technologies for chemistry. *Chemical Reviews*, 117(12):7673–7761. PMID: 28475312.

Krohn, J., Beyleveld, G., and Bassens, A. (2019). *Deep learning illustrated: a visual, interactive guide to artificial intelligence*. Addison-Wesley Professional.

Kumar, S. (2017). A survey of deep learning methods for relation extraction. *arXiv preprint arXiv:1705.03645*.

Lauriola, I., Lavelli, A., and Aiolli, F. (2022). An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470:443–456.

Lee, J. H., Lee, M., and Min, K. (2023). Natural language processing techniques for advancing materials discovery: A short review. *International Journal of Precision Engineering and Manufacturing-Green Technology*, pages 1–13.

Lee, J. Y., Dernoncourt, F., and Szolovits, P. (2017). Transfer learning for named-entity recognition with neural networks. *arXiv preprint arXiv:1705.06273*.

Li, J., Pradhan, B., Gaur, S., and Thomas, J. (2019a). Predictions and strategies learned from machine learning to develop high-performing perovskite solar cells. *Advanced Energy Materials*, 9(46):1901891.

Li, R., Fu, D., Shi, C., Huang, Z., and Lu, G. (2025). Efficient llms training and inference: An introduction. *IEEE Access*, 13:32944–32970.

Li, X., Xie, Q., Jiang, J., Zhou, Y., and Huang, L. (2019b). Identifying and monitoring the development trends of emerging technologies using patent analysis and twitter data mining: The case of perovskite solar cell technology. *Technological Forecasting and Social Change*, 146:687–705.

Liddy, E. D. (2001). Natural language processing.

Lin, Y.-J., Dai, H.-J., Zhang, Y.-C., Wu, C.-Y., Chang, Y.-C., Lu, P.-J., Huang, C.-J., Wang, Y.-T., Hsieh, H.-M., Chao, K.-S., Liu, T.-W., Chang, I.-S., Yang, Y.-H. C., Wang, T.-H., Liu, K.-J., Chen, L.-T., and Yang, S.-F. (2020). Cancer registry information extraction via transfer learning. In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 201–208.

Liu, R. and McKie, J. (2021). `https://pymupdf.readthedocs.io/en/latest/`.

Liu, S., Lu, Y., Yu, C., Li, J., Luo, R., Guo, R., Liang, H., Jia, X., Guo, X., Wang, Y.-D., et al. (2024). Triple-junction solar cells with cyanate in ultrawide bandgap perovskites. *Nature*, pages 1–3.

Liu, X., Sun, P., Chen, S., Zhang, L., Dong, P., You, H., Zhang, Y., Yan, C., Chu, X., and Zhang, T.-y. (2025). Perovskite-llm: Knowledge-enhanced large language models for perovskite solar cell research. *arXiv preprint arXiv:2502.12669*.

Loper, E. and Bird, S. (2002). Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Makino, K., Kuniyoshi, F., Ozawa, J., and Miwa, M. (2022). Extracting and analyzing inorganic material synthesis procedures in the literature. *IEEE Access*, 10:31524–31537.

Manjarres, A. V., Sandoval, L. G. M., and Suárez, M. S. (2018). Data mining techniques applied in educational environments. *Digital Education Review*, (33):235–266.

Mannai, M., Karâa, W. B. A., and Ghezala, H. H. B. (2017). Information extraction approaches: A survey. In *Information and communication technology: proceedings of ICICT 2016*, pages 289–297. Springer.

Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30.

McKinney, W. (2022). Pandas, https://pandas.pydata.org/.

Microsoft (2023). Copilot [ai companion], https://copilot.microsoft.com.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

MonkeyLearn (2021). Text classification with machine learning nlp, https://monkeylearn.com/text-classification/.

Moradi, H., Ahmadi, F., and Feizi Derakhshi, M. R. (2017). A hybrid approach for persian named entity recognition. *Iranian Journal of Science and Technology, Transactions A: Science*, 41.

Munzert, S., Rubba, C., Meißner, P., and Nyhuis, D. (2015). *Automated data collection with R: A practical guide to web scraping and text mining*, volume 1. Wiley West Sussex.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nagy, Z. (2018). *Regex quick syntax reference: understanding and using regular expressions*. Apress.

Naseer, S., Ghafoor, M. M., bin Khalid Alvi, S., Kiran, A., Rahmand, S. U., Murtazae, G., and Murtaza, G. (2021). Named entity recognition (ner) in nlp techniques, tools accuracy and performance. *Pakistan Journal of Multidisciplinary Research*, 2(2):293–308.

National Laboratory of the Rockies (2026). Best research-cell efficiency chart, https://www.nlr.gov/pv/cell-efficiency.

Neumann, M., King, D., Beltagy, I., and Ammar, W. (2019). Scispacy: fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP workshop and shared task*, pages 319–327.

Nguyen, M.-T., Phan, V.-A., Linh, L. T., Son, N. H., Dung, L. T., Hirano, M., and Hotta, H. (2020). Transfer learning for information extraction with limited data. In *Computational Linguistics*, pages 469–482.

Odabaşı, Ç. and Yıldırım, R. (2019). Performance analysis of perovskite solar cells in 2013–2018 using machine-learning tools. *Nano Energy*, 56:770–791.

OpenAI (2022). Chatgpt [large language model], `https://openai.com/chatgpt`.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Patwardhan, N., Marrone, S., and Sansone, C. (2023). Transformers in the real world: A survey on nlp applications. *Information*, 14(4):242.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Piskorski, J. and Yangarber, R. (2012). Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, pages 23–49. Springer.

Porto, J. P. and Merino, M. (2015). DEFINICIÓN DE PIPELINE, https://definicion.de/pipeline/.

Project, S. (2026). Selenium webdriver. https://www.selenium.dev/documentation/. Versión 4.20.0, consultado en mayo 2026.

Python Software Foundation (2021). Regular expression operations, https://docs.python.org/3/library/re.html.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A python natural language processing toolkit for many human languages.

Rabiee, N. (2023). Artificial intelligence assisted with designing metal-organic frameworks (mofs). *Clinical and Translational Discovery*, 3(3):e207.

Reitz, K. (2026). Requests: Http for humans. https://docs.python-requests.org/. Versión 2.31.0, consultado en mayo 2026.

Richardson, L. (2026). Beautiful soup documentation. https://www.crummy.com/software/BeautifulSoup/. Versión 4.12.3, consultado en mayo 2026.

Riverbank Computing Limited, The Qt Company (2021). Pyqt5 reference guide, https://www.riverbankcomputing.com/static/Docs/PyQt5/.

Rodrigo, J. A. (2011). Validación de modelos predictivos:cross-validation, oneleaveout, available under a attribution 4.0 international (cc by 4.0) , https://www.cienciadedatos.net/documentos/30_cross-validation_oneleaveout_bootstrap.

Rojas, V. M. and Córdova, R. S. (2021). Celdas solares de perovskita como alternativa para la electrificación rural del perú. *Social Innova Sciences, Revista de Ciencia Sociales*, 2.

Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway. http://hdl.handle.net/10379/15463.

S. REN21 (2025). RENEWABLES 2025, GLOBAL STATUS REPORT, https://www.ren21.net/gsr-2025/global_overview/.

Sabha, S. U., Assad, A., Din, N. M. U., and Bhat, M. R. (2024). From scratch or pretrained? an in-depth analysis of deep learning approaches with limited data. *International Journal of System Assurance Engineering and Management*, pages 1–10.

Schröder, C. and Niekler, A. (2020). A survey of active learning for text classification using deep neural networks.

Seow, W. L., Chaturvedi, I., Hogarth, A., Mao, R., and Cambria, E. (2025). A review of named entity recognition: from learning methods to modelling paradigms and tasks. *Artificial Intelligence Review*, 58(10):315.

Shabih, S., Koch, C. T., Jablonka, K. M., and Marquez, J. A. (2025). Automated data extraction from solar cell literature using large language models. In *AI for Accelerated Materials Design*.

Shah, N., Shah, A., Leung, P., Khan, S., Sun, K., Zhu, X., and Liao, Q. (2023). A review of third generation solar cells. *Processes*, 11(6):1852.

Shelar, H., Kaur, G., Heda, N., and Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3):324–337.

Singh, S. (2018). Natural language processing for information extraction. *arXiv preprint arXiv:1807.02383*.

Singrodia, V., Mitra, A., and Paul, S. (2019). A review on web scrapping and its applications. In *2019 international conference on computer communication and informatics (ICCCI)*, pages 1–6. IEEE.

Sun, M., Jiao, Z., Wang, P., Li, X., and Yuan, G. (2025). Strategies and methods for upscaling perovskite solar cell fabrication from lab-scale to commercial-area fabrication. *Molecules*, 30(10):2221.

Swain, M. C. and Cole, J. M. (2016). Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of chemical information and modeling*, 56(10):1894–1904.

Tahir, M. B., Rafique, M., Rafique, M. S., Nawaz, T., Rizwan, M., and Tanveer, M. (2020). Photocatalytic nanomaterials for degradation of organic pollutants and heavy metals. In *Nanotechnology and photocatalysis for environmental applications*, pages 119–138. Elsevier.

Tao, Q., Xu, P., Li, M., and Lu, W. (2021). Machine learning for perovskite materials design and discovery. *npj Computational Materials*, 7(23).

Thankappan, A., Thomas, S., and Wang, Y. (2025). *Perovskite photovoltaics: basic to advanced concepts and implementation*. Elsevier.

Thilakaratne, M., Falkner, K., and Atapattu, T. (2019). A systematic review on literature-based discovery: General overview, methodology, & statistical analysis. *ACM Comput. Surv.*, 52(6).

Tinn, R., Cheng, H., Gu, Y., Usuyama, N., Liu, X., Naumann, T., Gao, J., and Poon, H. (2023). Fine-tuning large neural language models for biomedical natural language processing. *Patterns*, 4(4).

Tonui, P., Oseni, S. O., Sharma, G., Yan, Q., and Tessema Mola, G. (2018). Perovskites photovoltaic solar cells: An overview of current status. *Renewable and Sustainable Energy Reviews*, 91:1025–1044.

Trewartha, A., Walker, N., Huo, H., Lee, S., Cruse, K., Dagdelen, J., Dunn, A., Persson, K. A., Ceder, G., and Jain, A. (2022). Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. *Patterns*, 3(4):100488.

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vélez, J., Sepúlveda, A., Botero, M., Otálora, C., and Cristian, C. (2023). Absorber layer thickness as a new feature in statistical learning tools of perovskite solar cells. *Journal of Applied Research and Technology*, 21.

Vepreva, A., Razlivina, J., Eremeyeva, M., Gubina, N., Orlova, A., Dmitrenko, A., Xenia, K., Jyakhwo, S., Vasilev, N., Sarkisyan, A., et al. (2026). Chemx: A collection of chemistry datasets for benchmarking automated information extraction. *Advances in Neural Information Processing Systems*, 38.

Vilcot, J.-P., Ayachi, B., Aviles, T., and Miska, P. (2017). Full sputtering deposition of thin film solar cells: a way of achieving high efficiency sustainable tandem cells? *Journal of Electronic Materials*, 46(11):6523–6527.

Wahed, M. A., Alzboon, M. S., Alqaraleh, M., Ayman, J., Al-Batah, M., and Bader, A. F. (2024). Automating web data collection: Challenges, solutions, and python-based strategies for effective web scraping. In *2024 7th International Conference on Internet Applications, Protocols, and Services (NETAPPS)*, pages 1–6. IEEE.

Wang, H., Li, J., Wu, H., Hovy, E., and Sun, Y. (2023). Pre-trained language models and their applications. *Engineering*, 25:51–65.

Wang, H., Qin, K., Zakari, R. Y., Lu, G., and Yin, J. (2022). Deep neural network-based relation extraction: an overview. *Neural Computing and Applications*, 34(6):4781–4801.

Wankmüller, S. (2022). Introduction to neural transfer learning with transformers for social science text analysis. *Sociological Methods & Research*, 0(0):00491241221134527.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020a). Transformers: State-of-the-art natural language processing. In Liu, Q. and Schlangen, D., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020b). Huggingface's transformers: State-of-the-art natural language processing.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wu, X.-K., Chen, M., Li, W., Wang, R., Lu, L., Liu, J., Hwang, K., Hao, Y., Pan, Y., Meng, Q., et al. (2025). Llm fine-tuning: Concepts, opportunities, and challenges. *Big Data and Cognitive*

Computing, 9(4):87.

Xie, T., Wan, Y., Zhou, Y., Huang, W., Liu, Y., Linghu, Q., Wang, S., Kit, C., Grazian, C., Zhang, W., and Hoex, B. (2024). Creation of a structured solar cell material dataset and performance prediction using large language models. *Patterns*, 5(5):100955.

Yang, Y., Huang, H., Yan, L., Cui, P., Lan, Z., Sun, C., Du, S., Wang, X., Yao, C., Qu, S., Zhang, Q., Wang, M., Zhao, X., and Li, M. (2024). Compatible soft-templated deposition and surface molecular bridge construction of sno2 enable air-fabricated perovskite solar cells with efficiency exceeding 25.7%. *Advanced Energy Materials*, page 2400416.

Yang, Y., Wu, Z., Yang, Y., Lian, S., Guo, F., and Wang, Z. (2022). A survey of information extraction based on deep learning. *Applied Sciences*, 12(19):9691.

Yao, J. and Yuan, B. (2024). Optimization strategies for deep learning models in natural language processing. *Journal of Theory and Practice of Engineering Science*, 4:80–87.

Yin, H., Xu, M., Luo, Z., Bi, X., Li, J., Zhang, S., and Wang, X. (2024). Machine learning for membrane design and discovery. *Green Energy Environment*, 9(1):54–70.

Yu, H., Fan, L., Li, L., Zhou, J., Ma, Z., Xian, L., Hua, W., He, S., Jin, M., Zhang, Y., Gandhi, A., and Ma, X. (2024). Large language models in biomedical and health informatics: A bibliometric review.

Yuan, M. (2022). Transfer learning in natural language processing through interactive feedback.

Zaman, G., Mahdin, H., Hussain, K., and Rahman, A. (2020). Information extraction from semi and unstructured data sources: A systematic literature review. *ICIC Express Letters*, 14(6):593–603.

Zhang, D., Li, B., Hang, P., Xie, J., Yao, Y., Kan, C., Yu, X., Zhang, Y., and Yang, D. (2024). Mitigated front contact energy barrier for efficient and stable perovskite solar cells", journal =.[en]ergy environ. sci.

Zhang, Y. and Yang, Q. (2017). An overview of multi-task learning. *National Science Review*, 5(1):30–43.

Zhao, Z., Alzubaidi, L., Zhang, J., Duan, Y., and Gu, Y. (2024). A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, 242:122807.

Zhu, X., Li, Y., and Gu, N. (2023). Application of artificial intelligence in the exploration and optimization of biomedical nanomaterials. *Nano Biomedicine and Engineering*, 15(3):342–353.

Apéndices

Apéndice A. Descripción detallada del proceso de generación del conjunto de datos etiquetados

A.1 Proceso de generación del primer versión del Dataset para la tarea NER

Para obtener la primer versión de los datos etiquetados para la tarea NER, se integran diferentes módulos y funciones:

- Un módulo que usa la librería PyMuPDF (Liu and McKie, 2021) para extraer en texto plano la información de los PDFs. El Script extrae y segmenta el texto del PDF en sus secciones más representativas.

- Un módulo que usa la librería *Beautiful Soup*, el conocimiento de la estructura HTML y el mecanismo de verificación de los documentos usado en el módulo de desarrollado en la anterior Sección 5.1.2, para extraer y segmentar el texto del documento HTML de una lista de revistas/editoriales (*Jornals*) web en sus secciones más representativas.

- Una serie de funciones para la definición de las nuevas entidades usando la librería *spaCy*. Aunque la librería cuenta con varios modelos preentrenados con una serie de entidades nombradas ya entrenadas, esta permite la definición y uso de nuevas entidades nombradas sin afinar (aplicar *Fine Tuning*) el modelo pre-entrenado usado. Para esto hay dos formas principales: el primero creando una lista con las palabras asociadas a las entidades y un segundo

creando patrones con una forma modificada de expresiones regulares para reconoces las palabras asociadas a las entidades nombradas.

El procedimiento es aplicado de forma secuencial para obtener el primer archivo JSON del *Dataset*:

1. Usar el sistema de archivos e iterar sobre cada archivo en bloques, para generar con módulos un diccionario de python con secciones del texto. Se extrae y segmenta la información de secciones como, el Abstract, Materials and Methods, Results and Discussion, Conclusions, etc, donde se espera encontrar información sobre las entidades nombradas (Tabla 7).

2. Con el diccionario de secciones y la librería *spaCy*, se extraen oraciones de forma automática. Durante la extracción de las oraciones, se seleccionan solo las que se consideren con sentido y útiles para entrenar el modelo, para esto se verifica que tengan: Un verbo principal, un sujeto o sustantivo dependiente del verbo, entre otros filtros.

3. Al llegar al último bloque de artículos se limpian todas las oraciones del archivo y elimina las oraciones repetidas, pasando de 151473 a 149181 oraciones. Se genera el archivo de registro con la información resumida de las oraciones eliminadas, al inicio y al final de la limpieza.

4. Las 149181 oraciones se dividen y guardan en grupos de máximo 10000, creando N archivos. Luego se definen la nuevas entidades usando los dos métodos de *spaCy* descritos anteriormente, usando patrones y listas de entidades ($Entity-List.xlsx$). Se obtienen los datos etiquetados de cada archivo y se organizan en los archivos.

5. Se obtiene la primera version del *Dataset* al agrupar los archivos JSON (*TRAINHTML.json*), y se actualiza el archivo de registro con la información de los nombres de los archivos generados.

A.2 Proceso de depuración del Dataset para la tarea NER

Para la Depuración del *Dataset* se desarrolla un modulo que mediante diferentes funciones permite:

- Revisar si hay solapamientos de indices e indices incorrectos. Esta función revisa, corrige y si es necesario le pregunta al usuario si desea borrar la anotación.
- Balancear el conjunto entre, oraciones con entidades y oraciones sin entidades, para un tamaño de datos requerido. Esta función permite decidir el porcentaje de oraciones en cada grupo y la máxima cantidad de oraciones que se van a balancear.
- El usuario puede seleccionar si desea borrar o aceptar las oraciones con sus entidades. Esta función itera cada oración y espera a que el usuario decida antes de pasar a la siguiente oración.
- Revisar que no existan oraciones duplicadas.
- Llamar una GUI, una app que funciona con una interfase gráfica que permite corregir, agregar o borrar manualmente las entidades de cada oración. Esta GUI fue originalmente desarrollada en nuestro trabajo previo (Jiménez-Díaz et al., 2022), ahora actualizada y modificada para alinearse a las nuevas necesidades. El Apéndice B describe de su funcionamiento.

- Revisar que las entidades estén alineadas con la tokenización que aplica *spaCy*. Esta función es necesaria aplicarla almenos una vez al final, debido al uso de las otras funciones, anotación manual y a las no raras inconsistencias que puede tener la librería al separar las palabras (tokenizar) en las oraciones, causada por caracteres invisibles, saltos de línea, puntuación poco usual o por la forma que la librería hace la tokenización, como una combinación de reglas lingüísticas y modelos estadísticos (Honnibal et al., 2026b). La función toma el conjunto de datos etiquetados, crea un modelo *spaCy* en blanco, agrega casos especiales de tokenización y genera una simulación para cada oración de como va a ser separada por tokenizador y registra en un TXT las entidades con sus oraciones a corregir.

A.3 Proceso de conversión y depuración del Dataset para la tarea RE

Mediante un sistema semiautomático se construye el conjunto de datos etiquetados utilizado para el entrenamiento, evacuación y validación del modelo extracción de relaciones (RE). El sistema sigue la secuencia:

- Combinar en el *Dataset* las entidades nombradas asociadas a los parámetros de desempeño con las asociadas a los valores tomados por los parámetros de desempeño. Se pasa de 16 a 11 etiquetas de entidades nombradas.

- Mediante un diccionario se definen los pares-entidad y la clase de relación que puede tener, se crean relaciones automáticas con un formato simple para la posterior corrección manual. Se consideran los pares-entidad que presentaban una relación positiva y los pares-entidad sin relación con el propósito de proporcionar al modelo ejemplos negativos y evitar que el

aprendizaje se restrinja exclusivamente a combinaciones relacionales. En la Tabla 8 se ven las relaciones positivas definidas mediante el esquema establecido en un diccionario, como se ve a continuación.

Code Listing 7.1. Diccionario para definir las clases usadas para las relaciones

```
{
        # Relaciones estructurales / de uso ---
        ("HTL", "PTF"): "USED_WITH",
        ("ETL", "PTF"): "USED_WITH",
        ("ETL", "HTL"): "USED_WITH",
        # Relaciones de desempeno  ---
        ("PTF", "VOC"): "HAS_VALUE",
        ("HTL", "VOC"): "HAS_VALUE",
        ("ETL", "VOC"): "HAS_VALUE",
        ("PTF", "PCE"): "HAS_VALUE",
        ("HTL", "PCE"): "HAS_VALUE",
        ("ETL", "PCE"): "HAS_VALUE",
        ("PTF", "JSC"): "HAS_VALUE",
        ("HTL", "JSC"): "HAS_VALUE",
        ("ETL", "JSC"): "HAS_VALUE",
        ("PTF", "FF") : "HAS_VALUE",
        ("HTL", "FF") : "HAS_VALUE",
        ("ETL", "FF") : "HAS_VALUE",
        # Relaciones de proceso / fabricacion ---
        ("HTL","MTDS"): "PREPARED_WITH",
        ("ETL","MTDS"): "PREPARED_WITH",
        ("PTF","MTDS"): "PREPARED_WITH",
        # Condiciones de proceso ---
        ("MTDS", "ENMTDS"): "PERFORMED_UNDER_CONDITION",
        ("MTDS", "VMTDS"): "PERFORMED_UNDER_CONDITION",
        ("MTDS", "NMTDS"): "PERFORMED_UNDER_CONDITION",
        ("VMTDS", "NMTDS"): "PERFORMED_UNDER_CONDITION"
}
```

- Para corregir de forma semi-manual las relaciones creadas, se implementa un sistema que permite decidir si borrar, cambiar o aceptar cada par entidades-relación que contiene el conjunto de datos. El sistema muestra todos los pares, pero solo se somete a revisión y depuración los pares correspondientes a las combinaciones contempladas.

Durante la etapa de depuración, cuando una relación generada automáticamente no correspondía con la información del texto, esta era corregida. Por su parte, los pares que no corresponden a alguna de las relaciones válidas, definidas dentro del esquema, se conservaron con la etiqueta "*NO_RELATION*". Por tanto, esta clase cumple la función de representar ejemplos negativos dentro del problema de clasificación, aunque su generación presenta un componente determinista asociado al esquema de relaciones empleado.

Apéndice B. API para la corrección de conjuntos de datos en formato JSON para el entrenamiento, evaluación y validación de modelos NER de spaCy

La API del scripts *GUI.py* escrito con "*PyQt5*" (Riverbank Computing Limited, The Qt Company, 2021) y "*Qt-designer*" (Herrmann, 2021) que permiten la generación de "*GUIs*" en lenguaje python, que fue inicalmente desarrollada en el proyecto previo (Jiménez-Díaz et al., 2022) y utilizada en este trabajo para apoyar el proceso de la Sección 5.2.

Esta API consta de unas ventanas sencillas y de la impresión en la terminal de linea de la estructura tokenizada de la oración, que permiten corregir un conjunto de datos en formato .JSON.

La primera ventana o ventana inicial que se ve en la Figura 30, permite ingresar el archivo .JSON estructurado como el del ejemplo de la Figura 16 de la sección 5.4 y un archivo nuevo o uno modificado en formato .JSON con la misma estructura como salida.

Figura 30

Ventana inicial de la API.



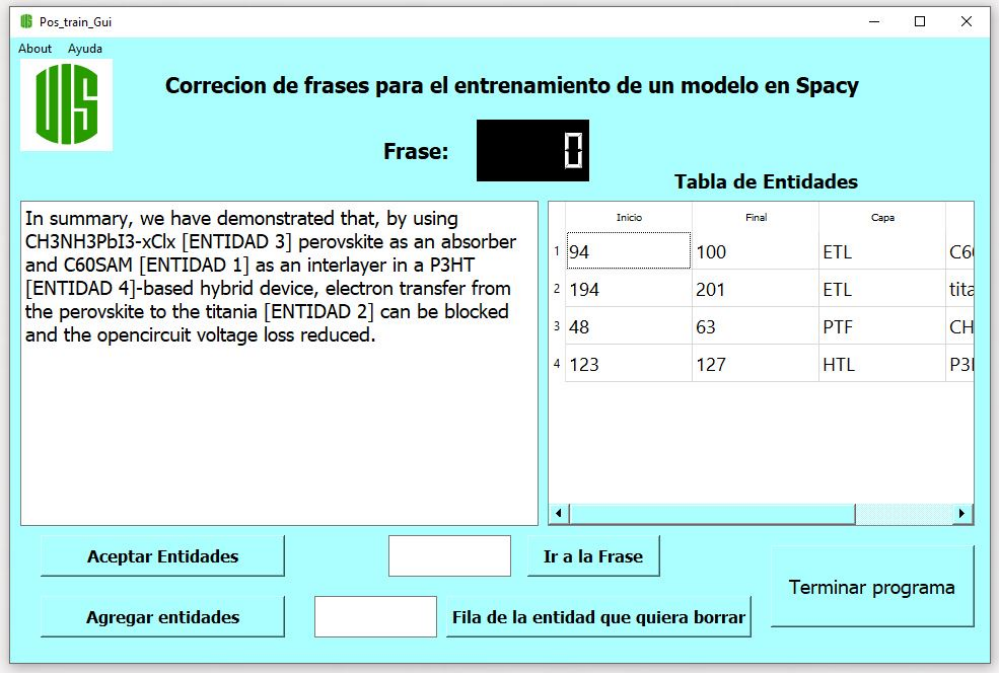
Nota. Ventana usada para ingresar el archivo con los datos a corregir y ingresar el archivo de salida a usar.

En la segunda ventana o la ventana principal que se ve en la Figura 31, se realizan las modificaciones usando 5 botones: dos para aceptar o agregar nuevas entidades nombradas, uno para saltar a una oración en particular, uno para eliminar una entidad y otro para salir de la API. La segunda ventana tiene una casilla de visualización donde se ve la oración que se esta trabajando, una casilla con una tabla que muestra las entidades nombradas y su posición en el texto, y la casilla con el numero de oración que se esta trabajando.

Por último la ventana secundaria o tercera ventana que se ve en la Figura 32, se usa para buscar e ingresar la entidad nombrada y el nombre de entidad nombrada que no se reconoció automáticamente.

Figura 31

Ventana principal de la API.



Nota. Ventana usada realizar la corrección visual y manual del archivo de datos.

Figura 32

Ventana secundaria de la API.



Nota. Ventana usada para buscar la posición e ingresar las nuevas entidades nombradas.

Apéndice C. Proceso de selección, optimización y validación del modelo aplicado para el entrenamiento de un modelo *Deep Learning* usando aprendizaje supervisado en Python.

Los detalles del proceso mostrado en la Figura 20 se describen a continuación.

1. ***Hiperparámetros iniciales*:** Se asignan valores a los hiperparámetros, sean valores unitarios o una lista de valores para cada hiperparámetro.

2. ***Estrategia de optimización de Hiperparámmetros*:** Se aplica una estrategia de búsqueda para identificar la configuración de hiperparámetros que produce el mejor desempeño del modelo posible. En este trabajo se utiliza la Búsqueda en Cuadrícula (*Grid Search*), la cual evalúa sistemáticamente todas las combinaciones posibles generadas a partir de conjuntos predefinidos de valores para cada hiperparámetro.

3. *Entrenamiento del modelo "X"*: Se entrena el modelo utilizando la configuración actual de hiperparámetros, siguiendo el procedimiento descrito en la Subsección 3.12.2.

4. *Evaluación de validación de del modelo "X"*: El modelo entrenado se evalúa mediante las métricas descritas en la Subsección 3.12.6 sobre el conjunto de validación. Dependiendo de la estrategia empleada y de los resultados obtenidos, el proceso puede finalizar o continuar con una nueva iteración.

5. *Variación de Hiperparámetros*: Se actualizan los hiperparámetros según la estrategia de optimización utilizada, generalmente mediante la evaluación iterativa de diferentes configuraciones candidatas.

6. ***Modelo con desempeño idóneo*:** Se selecciona la configuración de hiperparámetros y el modelo que presentan el mejor desempeño según las métricas de evaluación obtenidas. Dependiendo de la estrategia, puede ser necesario aplicar varias veces el ciclo del punto 2.

7. ***Validación cruzada k-Fold*:** Se aplica la validación cruzada k-Fold descrita en la Subsección 3.12.7.2 sobre todo el conjunto de datos (*Dataset*) para evaluar la capacidad de generalización del modelo. Si los resultados no son satisfactorios, el proceso puede reiniciarse ajustando los hiperparámetros o seleccionando un modelo preentrenado diferente.

Apéndice D. Extracción automática de la información de tablas artículos científicos en formato PDF y HTML

Durante el trabajo y por experiencia del proyecto previo (Jiménez-Díaz et al., 2022), se observa que algunas entidades nombradas asociadas a los parámetros de desempeño de las celdas solares estan expresadas en las tablas y no en otras partes del texto del articulo. Por esto se decide trabajar de forma paralela al *Pipeline* principal con la extracción de tablas de PDFs y HTMLs. Se logra generar un Script *Table_extraction.py* y unas funciones en el script *SegmentacionHTML.py* que permiten extraer al menos un pequeño porcentaje de todas las tablas presentes en los 1307 artículos y guardarlos en formato EXCEL en la carpeta *TABLAS_EXT*. En total en la carpeta hay 376 archivos EXCEL y un TXT de registro.

El script *Table_extraction.py* usa la librería de libre acceso *Tabula* de python que es descrita como un contenedor simple de Python de tabula-java, que puede leer tablas desde un PDF y convertirlas en un DataFrame de *pandas* (McKinney, 2022) y también le permite convertir un archivo PDF en un archivo CSV, TSV o JSON (Ariga, 2021). Mediante el uso de expresiones regulares se encuentran los bloques de texto que contengan tablas, si existen toma los valores de la posición del bloque y los usa para extraer con la librería de *Tabula* la tabla, en caso de no poder encontrar la posición se aplica *Tabula* a todo el PDF. Y Las funciones en el script *SegmentacionHTML.py* usan la propia estructura de los archivos HTML para intentar detectar y extraer las tablas que pueda contener el articulo. Después de que se logre extraer alguna tabla del HTML o PDF, se realiza un acondicionamiento para obtener su titulo y comprobar que efectivamente se trate de una, comprobada se guardada en una hoja en un excel.

La funcionalidad de extracción de información desde tablas no fue integrada al *Pipeline* principal, debido a que aún no alcanza un desempeño suficientemente robusto. Entre las principales dificultades identificadas se encuentran la estructura de doble columna de numerosos archivos PDF y el almacenamiento de ciertas tablas en archivos web separados del documento HTML principal. En la Figura 33 se ve una tabla exitosamente extraída del HTML *HTMLs_620.html* y en la Figura 34 una tabla erróneamente extraída del PDF *PDFs_1207.pdf*.

Figura 33

Tabla exitosamente extraída del artículo HTMLs_620.html.

	A	B	C	D	E
1	**Devices**	**JSC (mA/cm2)**	**VOC (V)**	**FF (%)**	**η (%)**
2	C60/BCP	16.23	0.90	72.4	10.60
3	ZnO	—	—	—	—
4	5 nm-C60/ZnO	18.45	0.25	28.9	1.32
5	15 nm-C60/ZnO	19.41	0.91	62.0	10.93
6	30 nm-C60/ZnO	18.11	0.90	55.5	9.07
7					

Nota. Captura de pantalla del Excel generado a partir de la tabla extraída del documento.

Figura 34

Tabla erróneamente extraída del artículo PDFs_1207.pdf.

	A	B	C	D	E	F	G	H
1	**tionalization with**	**BBA SAM,**	**and**	**the**	**inverted**	**device with**	**this**	**func-**
2	tionalised interface	showed	better		stability	comparison	with	non-
3	functionalized interface device. Such results show that the stability of							
4	the perovskite film	and device	seems		not	correlated	with	the
5	hydrophilicity degree	of the	MOx		surface. Wang	et al. also	have	ob-
6	served that devices with BBA SAM showed moderate improvement in							
7	stability of inverted structure devices when tested in the dark (Wang							
8	et al., 2017). In other	work,	the	use	of phosphonic	acids	SAM	in
9	significant increase	in device	longevity		were	(Weerasingh		
10								

Nota. Captura de pantalla del Excel generado a partir de la tabla extraída del documento.

Apéndice E. Descripciones de las estructuras de datos usadas

E.1 Estructura de los datos del Dataset de entrenamiento para el modelo NER en JSON

$$[Sentence, \{entities : [[star, end, label], [star, end, label]]\}, ...]$$

- *Sentence*: oración original.
- *entities*: diccionario con clave "entities" que almacena un grupo de listas con componentes:
  - *start*: posición inicial de la entidad respecto a la oración.
  - *end*: posición final de la entidad respecto a la oración.
    - *label*: nombre o etiqueta asignada a la entidad nombrada.

E.2 Estructura de los datos del Dataset de entrenamiento para el modelo RE en JSON

$$[\{text : ...\texttt{<e1>}entity1\texttt{</e1>}...\texttt{<e2>}entity2\texttt{</e2>}..., label : Relation\}, ..]$$

- *text*: clave "text" para ingresar la oración.
- *<e#>...</e#>*: marcadores especiales de entidad (*entity markers*) para delimitar explícitamente los límites de las entidades objetivo dentro de cada secuencia de entrada. Para la tarea RE de este trabajo # solo toma los valores 1 y 2.
- *entity#*: entidad nombrada encontrada.
- *label*: clave "label" para ingresar la clase de relación encontrada entre dos entidades.
- *Relation*: nombre/etiqueta de la relación encontrada.

E.3 Estructura de los datos resultado de la inferencia del modelo NER en JSON

$$\begin{array}{l}
\{ \\
seccion : [\\
\quad \{seccion_text : text, windows : [list]\}, \\
\quad [\\
\quad Sentence, \{ \\
\quad\quad entities : [[star, end, label], [star, end, label]] \\
\quad\quad entities_seccion : [[star_s, end_s, label], [star_s, end_s, label]] \\
\quad\quad \}, ...], ...], ... \\
\}
\end{array}$$

seccion: diccionario que tiene como clave el nombre de la sección (ej. Abstarct), que almacena la extracción NER de la sección.

- *seccion_text*: diccionario con clave el "texto_nombre" de la sección que almacena un grupo de listas con componentes:
  - *text*: texto original de la sección.
  - *windows*: diccionario con clave "ventanas" que almacena una lista.
    - *list*: una lista con el grupo de ventanas generadas.
- *Sentence*: ventana de 2 oraciones.
- *entities*: diccionario con clave "entities" que almacena un grupo de listas con componentes:
  - *start*: posición inicial de la entidad respecto a la ventana de dos oraciones.
  - *end*: posición final de la entidad respecto a la ventana de dos oraciones.
  - *label*: nombre o etiqueta asignada a la entidad nombrada.
- *entities_seccion*: diccionario con clave "entities" que almacena un grupo de listas con componentes:
  - *start_s*: posición inicial de la entidad respecto a la sección.
  - *end_s*: posición final de la entidad respecto a la sección.
  - *label*: nombre o etiqueta asignada a la entidad nombrada.

E.4 Estructura de los datos resultado de la inferencia del modelo RE en JSON

{
seccion : [
{*seccion_text* : *text*, *windows* : [*list*]},
[
Sentence, {
text_market : ...<e1>*entity*1</e1>...<e2>*entity*2</e2>...,
relation : *relation_class*,
probability : *num*,
entities : [[*star*, *end*, *label*], [*star*, *end*, *label*]]
entities_seccion : [[*star_s*, *end_s*, *label*], [*star_s*, *end_s*, *label*]]
}, ...], ...], ...
}

seccion: diccionario que tiene como clave el nombre de la sección (ej. Abstarct), que almacena la extracción RE de la sección.

- *seccion_text*: diccionario con clave el "texto_nombre" de la sección que almacena un grupo de listas con componentes:
  - *text*: texto original de la sección.
  - *windows*: diccionario con clave "ventanas" que almacena una lista.
    - *list*: una lista con el grupo de ventanas generadas.
- *Sentence*: ventana de 2 oraciones.
- *text_market*: diccionario con clave "text_market" para ingresar el texto de la ventana de dos oraciones

- *<e#>...</e#>*: marcadores especiales de entidad (*entity markers*) para delimitar explícitamente los límites de las entidades objetivo dentro de cada secuencia de entrada. Para la tarea RE de este trabajo # solo toma los valores 1 y 2.
- *entity#*: entidad nombrada encontrada.
- *relation*: clave "relation" para ingresar la clase de relación encontrada entre dos entidades.
  relation_class: nombre/etiqueta de la relación encontrada.
- *probability*: clave "probability" para ingresar la probabilidad de la relación encontrada.
  num: Valor de probabilidad de la relación encontrada.
- *entities*: diccionario con clave "entities" que almacena un grupo de listas con componentes:
  - *start*: posición inicial de la entidad respecto a la ventana de dos oraciones.
  - *end*: posición final de la entidad respecto a la ventana de dos oraciones.
  - *label*: nombre o etiqueta asignada a la entidad nombrada.
- *entities_seccion*: diccionario con clave "entities" que almacena un grupo de listas con componentes:
  - *start_s*: posición inicial de la entidad respecto a la sección.
  - *end_s*: posición final de la entidad respecto a la sección.
  - *label*: nombre o etiqueta asignada a la entidad nombrada.

E.5 Estructura de los datos resultado del post-procesamiento en JSON

$$
\begin{aligned}
&\{\\
&seccion : \{\\
&\quad seccion_text : text,\\
&\quad Around_Tag : [\\
&\quad\quad \{\\
&\quad\quad device_id : num_id,\\
&\quad\quad structure : string,\\
&\quad\quad layers : [list_layers],\\
&\quad\quad methods : [list_methods],\\
&\quad\quad JSC : value_jsc,\\
&\quad\quad VOC : value_voc,\\
&\quad\quad PCE : value_pce,\\
&\quad\quad FF : value_ff,\\
&\quad\quad optimistic_score : \{dicc_opscore\},\\
&\quad\quad realistic_score : \{dicc_rescore\},\\
&\quad\quad device_summary : \{\\
&\quad\quad\quad layers : [[star, end, label_layers], [...]],\\
&\quad\quad\quad performance : [[star, end, label_performance], [...]],\\
&\quad\quad\quad methods : [[star, end, label_methods], [...]],\\
&\quad\quad\quad structure : [string, s_star, s_end, [entities_structure]],\\
&\quad\quad \}\}, ...], ...\\
&\quad Around_tag : [\{device_id : num_id, ...\}, ...], ...\},
\end{aligned}
$$

seccion : {....}, ...
}

seccion: diccionario que tiene como clave el nombre de la sección (ej. Abstarct), que almacena los devices que se encontraron dentro de la sección.

- *seccion_text*: diccionario con clave el "texto_nombre" de la sección que almacena un grupo de listas con componentes:
  - *text*: texto original de la sección.
- *Around_Tag*: diccionario con 3 posibles claves: "AROUND_PERFORMANCE", "AROUND_METHODS" y "AROUND_LAYERS", en estos se guardan los devices agrupados alrededor de las capas relacionadas al desempeño, los métodos o las capas, respectivamente:
  - *device_id*: diccionario con clave el "device_id" que guarda el numero de device del diccionario *Around_Tag*.
  - *num_id*: Valor numérico del device.
  - *structure*: diccionario con clave el "structure" que guarda la estructura producida por las capas de device del diccionario *Around_Tag*.
  - *string*: estructura producida por las capas de device del diccionario *Around_Tag*.
  - *layers*: diccionario con clave el "layers" con una lista de las capas agrupadas en el device del diccionario *Around_Tag*:
    - *list_layers*: lista de las capas agrupadas en el device del diccionario *Around_Tag*.
  - *methods*: diccionario con clave el "methods" con una lista de los métodos agrupados en el device del diccionario *Around_Tag*.
    - *list_methods*: lista de los métodos agrupados en el device del diccionario *Around_Tag*.
  - *JSC*: diccionario con clave el "JSC" que guarda el valor de la entidad jsc del device del diccionario *Around_Tag*.
  - *value_ jsc*: valor que toma el jsc del device del diccionario *Around_Tag*.
  - *VOC*: diccionario con clave el "VOC" que guarda el valor de la entidad voc del device del diccionario *Around_Tag*.
  - *value_ voc*: valor que toma el voc del device del diccionario *Around_Tag*.
  - *PCE*: diccionario con clave el "PCE" que guarda el valor de la entidad pce del device del diccionario *Around_Tag*.
  - *value_ pce*: valor que toma el pce del device del diccionario *Around_Tag*.
  - *FF*: diccionario con clave el "FF" que guarda el valor de la entidad ff del device del diccionario *Around_Tag*.
  - *value_ ff*: valor que toma el ff del device del diccionario *Around_Tag*.
  - *optimistic_score*: diccionario con clave el "optimistic_score" un diccionario de métricas de evaluación optimista (collapsed) del device del diccionario*Around_Tag*. Solo valida presencia global de la relación.
    - *dicc_opscore*: diccionario métricas de evaluación optimista del device del diccionario *Around_Tag*.
  - *realisticc_score*: diccionario con clave el "realistic_score" un diccionario de métricas de evaluación realista (extended) del device del diccionario*Around_Tag*. Valida presencia + coherencia posicional.

    * *dicc_rescore*: diccionario métricas de evaluación realista del device del diccionario *Around_Tag*.
- *device_summary*: diccionario con clave el "device_summary" un diccionario con el resument de las capas que hacen parte del device del diccionario *Around_Tag*.
    * *layers*: diccionario con clave el "layers" con la lista resume las capas agrupadas en el device del diccionario *Around_Tag*
    * *performance*: diccionario con clave el "performance" con la lista que resume las parámetros de desempeño agrupadas en el device del diccionario *Around_Tag*
    * *methods*: diccionario con clave el "methods" con la lista que resume los métodos agrupados en el device del diccionario *Around_Tag*.
        - *start*: posición inicial de la entidad respecto a la sección.
        - *end*: posición final de la entidad respecto a la sección.
        - *label_#*: nombre o etiqueta asignada a la entidad nombrada.
    * *structure*: diccionario con clave el "structure" con la lista que resume la estructura del device del diccionario *Around_Tag*.
        - *string*: estructura producida por las capas de device del diccionario *Around_Tag*.
        - *s_start*: posición inicial de la entidad respecto a la sección.
        - *s_end*: posición final de la entidad respecto a la sección.
        - *entities_structure*: lista resumen de las capas que hacen al estructura del device del diccionario *Around_Tag*.

Apéndice F. Descripción detallada del proceso de post-procesamiento basado en Ingeniería del Conocimiento

Mediante un módulo que trabaja por bloques de artículos, se usan una serie de funciones aplicadas a las entidades y relaciones obtenidas en la etapa anterior, los datos resultados de la fase de inferencia del modelo RE, la extracción de relaciones a nivel de ventanas de 2 oraciones guardados en la carpeta *DATA_RE*. La funciones trabajan a nivel de secciones:

- Se listan en orden de posición en el texto, la relaciones consignadas en los JSON que tengan un valor de probabilidad ofrecida por el modelo RE, de más del 50 %.
- Si la lista de relaciones no esta vacía, se aplican reglas de cercanía para crear otra lista de pares entidades-relación. Se busca en que oración esta cada entidad y luego se limita la distancia según las relaciones, a distancia de 1 o 2 oraciones. Dos entidades que estén en el diccionario de relaciones (Code Listing 7.1) y que cumplan estas reglas generan un pare entidades-relación. Para cada sección las dos listas se juntan en un solo conjunto global.
- Con el conjunto global se aplica un post-procesamiento. Dependiendo de la sección que se esta trabajando se generan nodos (entidades) núcleo para agrupar las demás relaciones a su alrededor creando el device. Se usan nuevamente reglas de distancia (máximo 400 caracteres de distancia) de modo que cada device creado ocupa una ventana de caracteres ubicada en una posición especifica en el texto de cada sección. En secciones como “Abstract”, “Discussion” o “Conclusions” se prioriza en orden la agrupación alrededor de, las entidades relacionadas a los parámetros de desempeño, las entidades relacionadas a capas y a las entidades

relacionadas a los métodos. En otras secciones como “Methods, “Experimental” o “Data”, la prioridad tiene un orden inverso.

- Se evalúa la coherencia de cada *device* comparando las entidades extraídas contra las relaciones identificadas por el modelo *MODEL_re*. Para ello se calculan dos métricas complementarias:

  - *Métrica optimista (collapsed):* mide la proporción de entidades cuyas relaciones pueden ser encontradas en el conjunto global de relaciones extraídas por el modelo. Esta métrica verifica únicamente la existencia de la relación, sin considerar si la entidad está asociada a una instancia específica del *device*.
  - *Métrica realista (extended):* mide la proporción de entidades cuyas relaciones no solo existen en el conjunto global de relaciones extraídas, sino que además se encuentran correctamente asociadas al *device* evaluado según su posición dentro del texto. Esta métrica incorpora restricciones espaciales y resulta más exigente al evaluar la consistencia estructural de la extracción.

  En ambos casos, valores cercanos a 1 indican una mayor concordancia entre las entidades extraídas y las relaciones inferidas por el modelo RE.

- Se busca, si existe, la estructura completa de la celda solar de perovskita. Debe estar separada por los delimitadores “ / ” ó “ | ” y contener almenos una entidad ya etiquetada. Para cada entidad relacionada con las capas, se busca hacia adelante y atrás por los deli-

mitadores, si se encuentran, se recorre el texto con los delimitadores hasta encontrar con expresiones regulares algún material asociado al sustrato o electrodo de la lista guardada en el JSON $sustratos - electrodos.json$. Una vez delimitada la estructura, se infiere con modelo $MODEL_ner$ las capas que pertenecen a la estructura y se crea una lista de listas para guardar la información de la estructura encontrada.

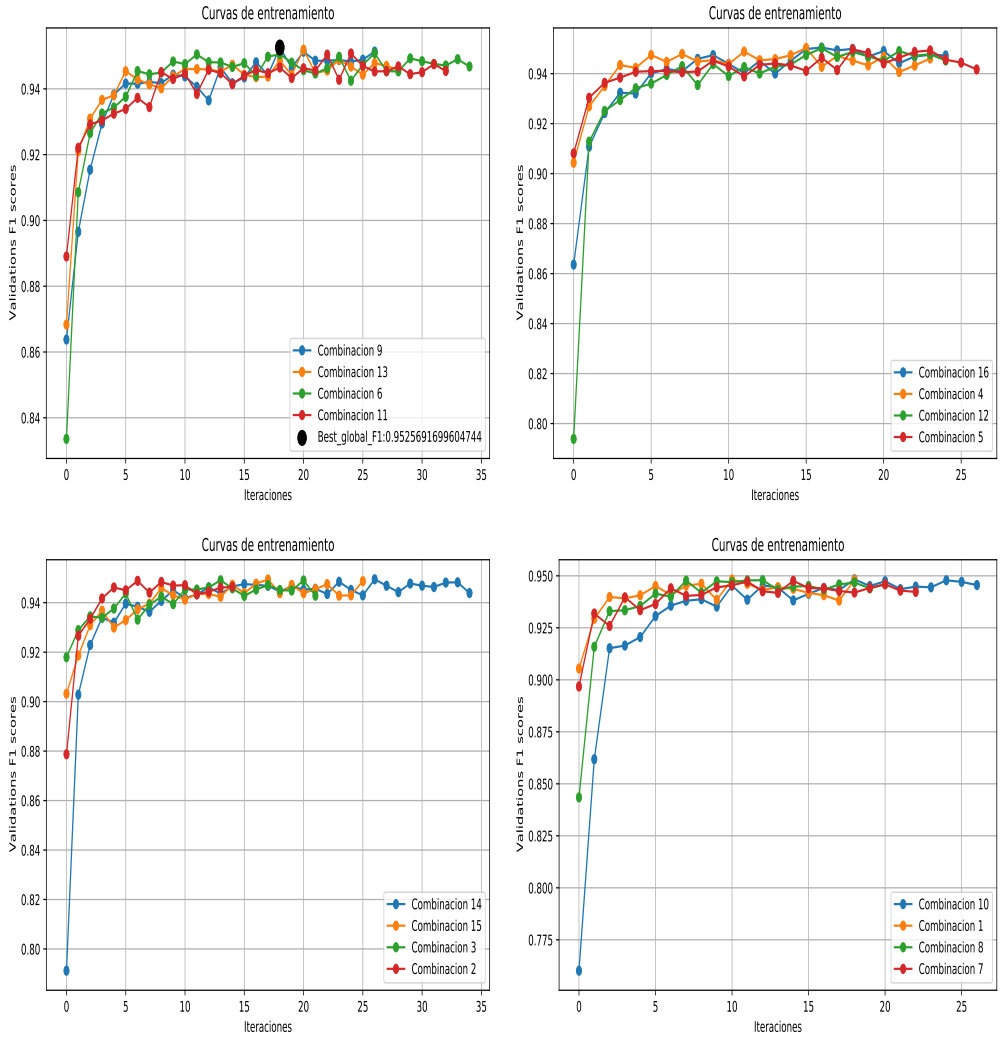
- Se integran la listas de estructuras encontradas a el grupo de devices. Cada estructura solo se puede integrar a un device, entonces se hacen todas los posibles pares device-estructura y luego solo de dejan los pares con menos distancia entre el centro de cada uno. Para ser un par device-estructura la posición de la estructura debe de estar dentro de la ventana del device o muy cerca de sus limites.

Apéndice G. Gráficas

G.1 Gráficas de las curvas de entrenamiento del proceso de creación del modelo NER

Figura 35

Curvas de entrenamiento de las 16 combinaciones evaluadas para el modelo NER.

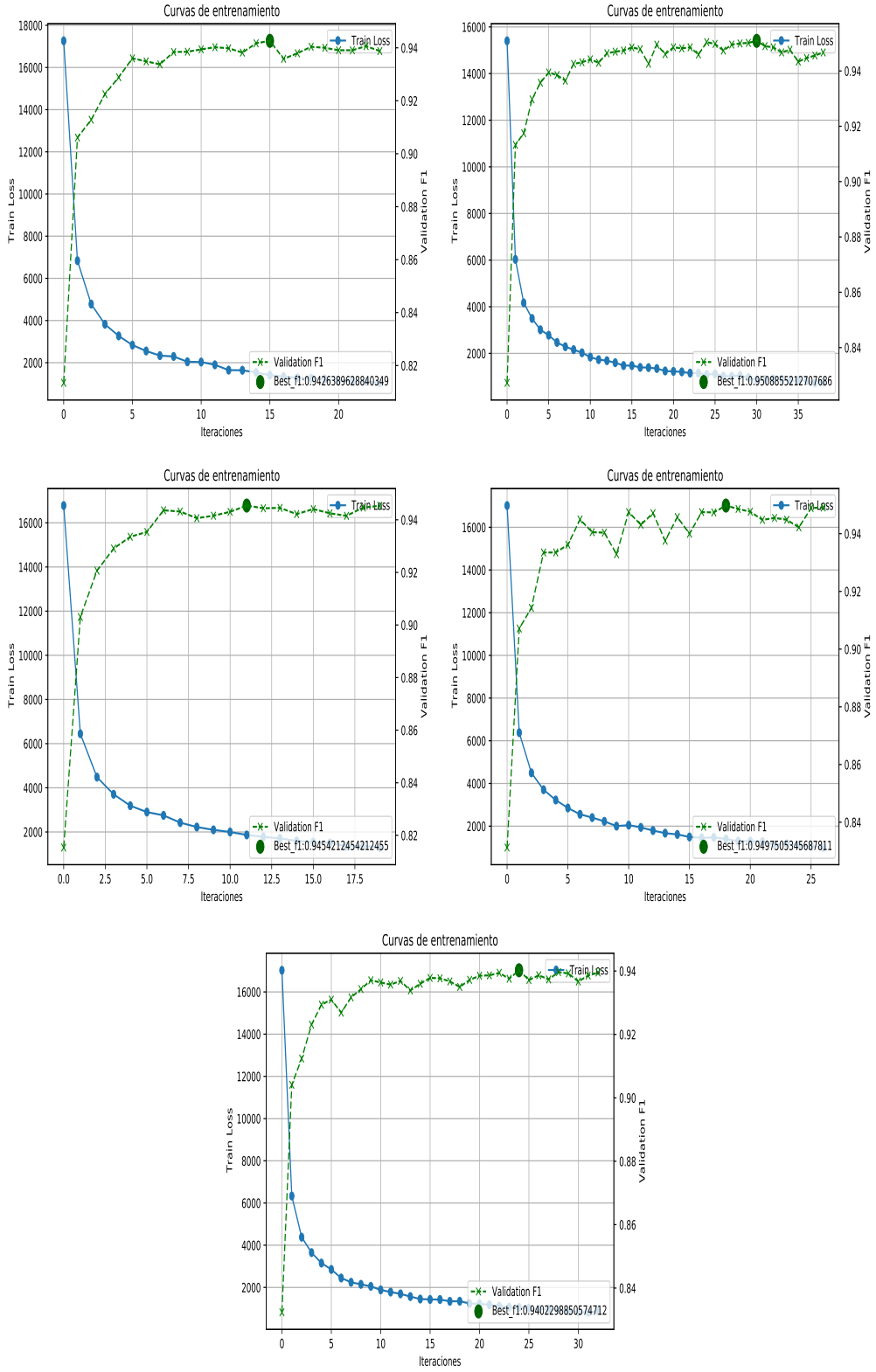


Nota. Las combinaciones fueron generadas mediante el procedimiento de búsqueda de hiperparámetros (*Grid Search*).

G.2 Gráficas de las curvas de entrenamiento del proceso de validación del modelo NER

Figura 36

Curvas de entrenamiento y pérdida (loss) obtenidas durante el proceso de validación cruzada por k-Fold, con $K = 5$.



Apéndice H. Tabla Resumen de los resultados de la Matriz de Hiperparámetros del proceso del modelo RE.

Tabla 40

Resumen de las métricas macro de evaluación de las 108 combinaciones evaluadas para el modelo RE.

$Matriz =$ [Per-device_batch size, learn_rate, weight_decay, dropout, Freeze_Layers]

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
0	[12, 2e-05, 0.01, 0.1, 0]	0.930	0.936	0.926	0.771	0.744	0.800	0.950	0:10:07.268484
1	[12, 2e-05, 0.01, 0.1, 3]	0.930	0.945	0.917	0.775	0.754	0.798	0.950	0:08:07.401586
3	[12, 2e-05, 0.01, 0.2, 0]	0.923	0.935	0.912	0.771	0.744	0.800	0.945	0:09:57.033187
4	[12, 2e-05, 0.01, 0.2, 3]	0.931	0.943	0.920	0.774	0.751	0.798	0.951	0:08:26.193033
5	[12, 2e-05, 0.01, 0.2, 6]	0.911	0.918	0.904	0.761	0.728	0.798	0.937	0:06:14.742661
6	[12, 2e-05, 0.01, 0.3, 0]	0.927	0.934	0.920	0.770	0.743	0.800	0.947	0:10:19.079673
7	[12, 2e-05, 0.01, 0.3, 3]	0.930	0.945	0.917	0.775	0.754	0.798	0.950	0:08:20.719615

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
8	[12, 2e-05, 0.01, 0.3, 6]	0.911	0.920	0.903	0.762	0.730	0.798	0.937	0:06:35.016741
9	[12, 2e-05, 0.05, 0.1, 0]	0.926	0.941	0.913	0.774	0.751	0.800	0.947	0:09:24.428330
10	[12, 2e-05, 0.05, 0.1, 3]	0.924	0.934	0.915	0.770	0.743	0.798	0.946	0:07:31.261821
11	[12, 2e-05, 0.05, 0.1, 6]	0.910	0.921	0.900	0.763	0.732	0.798	0.936	0:06:53.686425
12	[12, 2e-05, 0.05, 0.2, 0]	0.928	0.926	0.930	0.764	0.733	0.798	0.950	0:09:26.349550
13	[12, 2e-05, 0.05, 0.2, 3]	0.929	0.946	0.914	0.776	0.756	0.798	0.949	0:08:11.326647
14	[12, 2e-05, 0.05, 0.2, 6]	0.912	0.920	0.904	0.762	0.730	0.798	0.938	0:06:55.756927
15	[12, 2e-05, 0.05, 0.3, 0]	0.924	0.940	0.908	0.774	0.751	0.800	0.945	0:10:07.638475
16	[12, 2e-05, 0.05, 0.3, 3]	0.924	0.930	0.919	0.767	0.738	0.798	0.947	0:07:38.039707

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
17	[12, 2e-05, 0.05, 0.3, 6]	0.911	0.918	0.905	0.760	0.728	0.798	0.938	0:06:32.957057
18	[12, 3e-05, 0.01, 0.1, 0]	0.937	0.940	0.934	0.772	0.747	0.800	0.956	0:10:08.554146
19	[12, 3e-05, 0.01, 0.1, 3]	0.930	0.944	0.916	0.773	0.753	0.795	0.951	0:08:04.944942
20	[12, 3e-05, 0.01, 0.1, 6]	0.923	0.929	0.920	0.767	0.738	0.800	0.944	0:07:00.564238
21	[12, 3e-05, 0.01, 0.2, 0]	0.928	0.936	0.922	0.770	0.744	0.798	0.950	0:10:12.989291
22	[12, 3e-05, 0.01, 0.2, 3]	0.928	0.932	0.925	0.767	0.739	0.798	0.950	0:08:17.245785
23	[12, 3e-05, 0.01, 0.2, 6]	0.921	0.916	0.926	0.758	0.723	0.798	0.945	0:06:26.486446
24	[12, 3e-05, 0.01, 0.3, 0]	0.931	0.935	0.929	0.769	0.742	0.798	0.951	0:10:16.933106
25	[12, 3e-05, 0.01, 0.3, 3]	0.929	0.938	0.921	0.771	0.747	0.798	0.951	0:08:17.861240

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
26	[12, 3e-05, 0.01, 0.3, 6]	0.921	0.927	0.919	0.766	0.736	0.800	0.943	0:06:11.842507
27	[12, 3e-05, 0.05, 0.1, 0]	0.930	0.933	0.927	0.769	0.740	0.800	0.951	0:09:13.889625
28	[12, 3e-05, 0.05, 0.1, 3]	0.929	0.937	0.921	0.771	0.746	0.798	0.950	0:08:09.961428
29	[12, 3e-05, 0.05, 0.1, 6]	0.921	0.925	0.920	0.765	0.733	0.800	0.943	0:06:29.628421
30	[12, 3e-05, 0.05, 0.2, 0]	0.939	0.956	0.923	0.781	0.764	0.798	0.957	0:10:09.945213
31	[12, 3e-05, 0.05, 0.2, 3]	0.928	0.938	0.919	0.771	0.747	0.798	0.950	0:07:33.703329
32	[12, 3e-05, 0.05, 0.2, 6]	0.921	0.929	0.918	0.767	0.738	0.800	0.943	0:06:19.649468
33	[12, 3e-05, 0.05, 0.3, 0]	0.933	0.944	0.925	0.774	0.752	0.798	0.953	0:10:14.856074
34	[12, 3e-05, 0.05, 0.3, 3]	0.927	0.934	0.920	0.767	0.742	0.795	0.950	0:07:34.566731

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
35	[12, 3e-05, 0.05, 0.3, 6]	0.920	0.915	0.924	0.758	0.723	0.798	0.944	0:06:24.074900
36	[12, 5e-05, 0.01, 0.1, 0]	0.939	0.943	0.935	0.774	0.750	0.800	0.957	0:10:14.872875
37	[12, 5e-05, 0.01, 0.1, 3]	0.927	0.928	0.927	0.766	0.735	0.800	0.949	0:07:25.998191
38	[12, 5e-05, 0.01, 0.1, 6]	0.929	0.930	0.927	0.766	0.737	0.798	0.951	0:06:55.777686
39	[12, 5e-05, 0.01, 0.2, 0]	0.920	0.923	0.918	0.763	0.731	0.798	0.945	0:06:45.477561
40	[12, 5e-05, 0.01, 0.2, 3]	0.935	0.945	0.926	0.776	0.754	0.800	0.954	0:08:25.852184
41	[12, 5e-05, 0.01, 0.2, 6]	0.927	0.915	0.941	0.756	0.720	0.798	0.950	0:06:26.281978
42	[12, 5e-05, 0.01, 0.3, 0]	0.932	0.933	0.931	0.768	0.740	0.798	0.953	0:08:33.262488
43	[12, 5e-05, 0.01, 0.3, 3]	0.936	0.942	0.931	0.773	0.749	0.798	0.955	0:07:49.547486

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
44	[12, 5e-05, 0.01, 0.3, 6]	0.917	0.936	0.900	0.772	0.748	0.798	0.941	0:05:46.002236
45	[12, 5e-05, 0.05, 0.1, 0]	0.940	0.948	0.933	0.776	0.755	0.798	0.958	0:10:03.341902
46	[12, 5e-05, 0.05, 0.1, 3]	0.933	0.934	0.932	0.769	0.741	0.800	0.953	0:08:13.995867
47	[12, 5e-05, 0.05, 0.1, 6]	0.920	0.911	0.931	0.754	0.717	0.798	0.946	0:06:21.588897
48	[12, 5e-05, 0.05, 0.2, 0]	0.940	0.950	0.931	0.777	0.757	0.798	0.959	0:10:13.545265
49	[12, 5e-05, 0.05, 0.2, 3]	0.926	0.922	0.931	0.762	0.729	0.800	0.948	0:07:01.306074
50	[12, 5e-05, 0.05, 0.2, 6]	0.926	0.932	0.920	0.768	0.740	0.798	0.948	0:06:55.958762
51	[12, 5e-05, 0.05, 0.3, 0]	0.934	0.937	0.932	0.770	0.744	0.798	0.955	0:10:09.219611
52	[12, 5e-05, 0.05, 0.3, 3]	0.924	0.918	0.931	0.760	0.725	0.800	0.947	0:06:15.205808

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
53	[12, 5e-05, 0.05, 0.3, 6]	0.928	0.929	0.926	0.766	0.736	0.798	0.950	0:07:00.190256
54	[4, 2e-05, 0.01, 0.1, 0]	0.926	0.933	0.921	0.767	0.741	0.795	0.950	0:14:39.523314
55	[4, 2e-05, 0.01, 0.1, 3]	0.923	0.931	0.915	0.767	0.740	0.798	0.947	0:12:09.446876
56	[4, 2e-05, 0.01, 0.1, 6]	0.909	0.908	0.911	0.754	0.717	0.798	0.939	0:10:12.195025
57	[4, 2e-05, 0.01, 0.2, 0]	0.936	0.959	0.916	0.782	0.768	0.797	0.956	0:14:15.727811
58	[4, 2e-05, 0.01, 0.2, 3]	0.923	0.927	0.919	0.765	0.735	0.798	0.947	0:10:35.400830
59	[4, 2e-05, 0.01, 0.2, 6]	0.908	0.911	0.907	0.756	0.720	0.798	0.937	0:09:54.614979
60	[4, 2e-05, 0.01, 0.3, 0]	0.923	0.926	0.921	0.766	0.734	0.800	0.947	0:09:10.580095
61	[4, 2e-05, 0.01, 0.3, 3]	0.925	0.942	0.911	0.774	0.752	0.798	0.947	0:12:02.474721

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
62	[4, 2e-05, 0.01, 0.3, 6]	0.912	0.922	0.904	0.763	0.732	0.798	0.939	0:09:05.910589
63	[4, 2e-05, 0.05, 0.1, 0]	0.932	0.952	0.915	0.780	0.761	0.800	0.952	0:12:04.109676
64	[4, 2e-05, 0.05, 0.1, 3]	0.919	0.935	0.906	0.770	0.745	0.798	0.944	0:11:18.811132
65	[4, 2e-05, 0.05, 0.1, 6]	0.914	0.918	0.912	0.760	0.726	0.798	0.942	0:09:58.446903
66	[4, 2e-05, 0.05, 0.2, 0]	0.918	0.923	0.914	0.762	0.731	0.797	0.944	0:10:33.253576
67	[4, 2e-05, 0.05, 0.2, 3]	0.916	0.937	0.902	0.771	0.748	0.795	0.940	0:09:13.226486
68	[4, 2e-05, 0.05, 0.2, 6]	0.915	0.924	0.908	0.764	0.734	0.798	0.942	0:09:56.437444
69	[4, 2e-05, 0.05, 0.3, 0]	0.927	0.932	0.926	0.769	0.740	0.800	0.949	0:10:33.976633
70	[4, 2e-05, 0.05, 0.3, 3]	0.925	0.932	0.918	0.768	0.741	0.798	0.948	0:10:31.431035

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
71	[4, 2e-05, 0.05, 0.3, 6]	0.913	0.922	0.906	0.763	0.732	0.798	0.940	0:09:57.707325
72	[4, 3e-05, 0.01, 0.1, 0]	0.934	0.937	0.933	0.769	0.743	0.797	0.955	0:10:52.856766
73	[4, 3e-05, 0.01, 0.1, 3]	0.935	0.934	0.936	0.768	0.740	0.800	0.956	0:11:56.449563
74	[4, 3e-05, 0.01, 0.1, 6]	0.909	0.912	0.909	0.757	0.721	0.798	0.937	0:06:07.617302
75	[4, 3e-05, 0.01, 0.2, 0]	0.938	0.946	0.930	0.776	0.754	0.800	0.958	0:10:54.523410
76	[4, 3e-05, 0.01, 0.2, 3]	0.927	0.936	0.918	0.772	0.745	0.800	0.948	0:08:38.925963
77	[4, 3e-05, 0.01, 0.2, 6]	0.920	0.921	0.920	0.763	0.729	0.800	0.944	0:08:00.466085
78	[4, 3e-05, 0.01, 0.3, 0]	0.929	0.941	0.919	0.773	0.749	0.800	0.952	0:13:51.284057
79	[4, 3e-05, 0.01, 0.3, 3]	0.927	0.940	0.917	0.770	0.749	0.792	0.951	0:11:50.754612

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
80	[4, 3e-05, 0.01, 0.3, 6]	0.908	0.925	0.898	0.767	0.737	0.800	0.934	0:07:13.560257
81	[4, 3e-05, 0.05, 0.1, 0]	0.933	0.938	0.928	0.771	0.746	0.800	0.954	0:10:35.566251
82	[4, 3e-05, 0.05, 0.1, 3]	0.927	0.938	0.918	0.771	0.747	0.798	0.950	0:11:49.284202
83	[4, 3e-05, 0.05, 0.1, 6]	0.910	0.924	0.899	0.766	0.735	0.800	0.936	0:09:22.231328
84	[4, 3e-05, 0.05, 0.2, 0]	0.925	0.927	0.924	0.765	0.735	0.800	0.949	0:11:30.648779
85	[4, 3e-05, 0.05, 0.2, 3]	0.930	0.952	0.911	0.780	0.763	0.797	0.950	0:09:58.959605
86	[4, 3e-05, 0.05, 0.2, 6]	0.915	0.917	0.913	0.759	0.726	0.798	0.943	0:10:21.564642
87	[4, 3e-05, 0.05, 0.3, 0]	0.935	0.958	0.916	0.783	0.768	0.800	0.954	0:11:52.936143
88	[4, 3e-05, 0.05, 0.3, 3]	0.930	0.940	0.922	0.773	0.748	0.800	0.952	0:11:40.414303

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
89	[4, 3e-05, 0.05, 0.3, 6]	0.899	0.899	0.899	0.751	0.710	0.800	0.930	0:06:30.270055
90	[4, 5e-05, 0.01, 0.1, 0]	0.932	0.929	0.937	0.766	0.735	0.800	0.953	0:11:43.912382
91	[4, 5e-05, 0.01, 0.1, 3]	0.936	0.951	0.924	0.779	0.759	0.800	0.956	0:11:33.175975
92	[4, 5e-05, 0.01, 0.1, 6]	0.927	0.946	0.909	0.777	0.756	0.800	0.948	0:08:28.692029
93	[4, 5e-05, 0.01, 0.2, 0]	0.946	0.943	0.948	0.773	0.748	0.800	0.963	0:13:58.131899
94	[4, 5e-05, 0.01, 0.2, 3]	0.935	0.940	0.930	0.772	0.747	0.800	0.955	0:09:15.197439
95	[4, 5e-05, 0.01, 0.2, 6]	0.931	0.940	0.924	0.773	0.748	0.800	0.952	0:09:45.575279
96	[4, 5e-05, 0.01, 0.3, 0]	0.946	0.951	0.941	0.778	0.757	0.800	0.963	0:14:01.900242
97	[4, 5e-05, 0.01, 0.3, 3]	0.942	0.951	0.933	0.779	0.759	0.800	0.958	0:11:43.367203

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
98	[4, 5e-05, 0.01, 0.3, 6]	0.931	0.942	0.922	0.774	0.750	0.800	0.952	0:09:08.691903
99	[4, 5e-05, 0.05, 0.1, 0]	0.936	0.950	0.925	0.777	0.758	0.797	0.955	0:11:27.326239
100	[4, 5e-05, 0.05, 0.1, 3]	0.935	0.936	0.935	0.769	0.742	0.800	0.956	0:10:31.368937
101	[4, 5e-05, 0.05, 0.1, 6]	0.926	0.932	0.922	0.769	0.740	0.800	0.948	0:09:45.716504
102	[4, 5e-05, 0.05, 0.2, 0]	0.940	0.939	0.940	0.771	0.745	0.800	0.959	0:14:31.694085
103	[4, 5e-05, 0.05, 0.2, 3]	0.933	0.940	0.927	0.772	0.748	0.800	0.953	0:11:51.996551
104	[4, 5e-05, 0.05, 0.2, 6]	0.921	0.928	0.918	0.766	0.737	0.797	0.945	0:07:17.711506
105	[4, 5e-05, 0.05, 0.3, 0]	0.931	0.927	0.936	0.764	0.733	0.798	0.953	0:13:04.911437
106	[4, 5e-05, 0.05, 0.3, 3]	0.933	0.925	0.941	0.763	0.731	0.800	0.954	0:11:24.234907

		Macro-average_all			Macro-average_Rel				
#	**Matriz**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**Accuracy**	**Tiempo**
107	[4, 5e-05, 0.05, 0.3, 6]	0.922	0.924	0.922	0.763	0.731	0.798	0.947	0:08:17.891575

Nota. Las combinaciones fueron generadas mediante el procedimiento de búsqueda de hiperparámetros (*Grid Search*) y fueron guardadas en el EXCEL *Comparacion_Hiper_re.xlsx*. El macro-average_all calcula la métrica por clase, para todas las clases y luego promedia sin ponderar, y el macro-average_all calcula la métrica por clase, sin incluir la clase "NO_RELATON" y luego promedia sin ponderar.

Apéndice I. Organización de los archivos y Códigos

Con fines de trazabilidad y reproducibilidad del proceso experimental, se presenta una descripción general de las principales carpetas, archivos y recursos utilizados durante la investigación.

Tabla 41

Correspondencia entre las secciones metodológicas y las carpetas implementadas.

Carpeta	Descripción	Sección
Codes	Contiene los códigos usados	Todas
DATA_TRAIN	Contiene los archivos HTML y PDF del Dataset usado.	5.1 - 5.3, 5.8
DATA_TEST	Contiene los archivos PDF del Dataset usado para el test independiente	5.4.4
Experiments_ner	Contiene todos los experimentos desarrollados para obtener el modelo NER y el modelo *MODEL_ner*.	5.6
Experiments_re	Contiene todos los experimentos desarrollados para obtener el modelo RE y el modelo *MODEL_re*.	5.7
DATA_NER	Contiene los archivos resultado de la inferencia con el modelo NER.	5.8.1
DATA_RE	Contiene los archivos resultado de la inferencia con el modelo RE.	5.8.2
DATA_RESULTS	Contiene los archivos resultado del proceso de post-proceamiento.	5.9
Comparisons_devices	Contiene los archivos resultado del proceso de validación de los resultados.	5.10

Tabla 42

Correspondencia entre las secciones metodológicas y los principales archivos utilizados.

Archivo	Descripción	Sección
Perovskite_database_content_all.xlsx	Archivo del Database manual usado.	5.1
ARTopen_Perovskite.xlsx	Lista de artículos OA encontrados.	5.1
ARTopen_Perovskite_pdfs.xlsx	Lista de artículos OA encontrados solo en formato PDF.	5.1
TRAINHTML.txt	Oraciones generadas automóticamente.	5.4.1

Archivo	Descripción	Sección
TRAINHTML.json	Primer archivo de datos etiquetados para el modelo NER.	5.4.1
TRAINHTMLL_RS1_NER_RS.json	Archivo de datos etiquetados corregido para el modelo NER. Los arhcivos TRAINHTMLL_RS1_NER_RS_T.json y TRAINHTMLL_RS1_NER_RS_E.json, son divisiones de este archivo.	5.4.1
TRAINHTMLL_RS1_L_RE.json	Primer archivo de datos etiquetados para el modelo RE.	5.4.2
TRAINHTMLL_RS1_L_RE_C_llm.json	Archivo de datos etiquetados corregido para el modelo RE.	5.4.2
Test_NER_RS_L.json	Archivo de datos etiquetados corregido usado en el Test independiente del modelo NER.	5.4.4
Test_RE_C_llm.json	Archivo de datos etiquetados corregido usado en el Test independiente del modelo RE.	5.4.4
DATA_EXCEL_ner.json	Lista de entidades consignadas en el Database manual.	5.10.1
DATA_EXCEL.json	Grupos de dispositivos consignados en el Database manual.	5.10.2

Tabla 43

Correspondencia entre las secciones metodológicas y los módulos implementados.

Script	Descripción	Sección
Funtions.py	Diferentes funciones para todo el Pipeline.	Todas
Unpywall_web.py	Verificar el estado OA de los artículos.	5.1.1
HTMLsDOWNLOAD.py	Descarga y recolección de artículos científicos en formato HTML.	5.1.2
PDFLIST.py	Identifica los artículos con solo formato PDF.	5.1.2, 5.4.1
Sentences.py	Genera el primer archivo de datos etiquetados para NER.	5.4.1
Segmentation.py	Limpia y segmenta el texto de los artículos en PDF.	5.4.1
SegmentationHTML.py	Limpia y segmenta el texto de los artículos en HTML.	5.4.1
Depuration.py	Diferentes funciones usadas para la corrección y mejora, de los datos etiquetados de la tarea NER.	5.4.2
Gui.py	Interfas grafica para la correcion de datos para la tarea NER.	5.4.2

Script	Descripción	Sección
PreRe.py - PreRe_llms.py	Diferentes funciones usadas para la corrección, mejora y conversión, de los datos etiquetados de la tarea RE.	5.4.3
Training_ner.py	Entrenamiento de modelos de reconocimiento de entidades (NER).	5.6
Score_ner.py	Evaluación de modelos de reconocimiento de entidades (NER).	5.6
Kfold_training_ner.py	Validación cruzada Kfolds para el modelo NER.	5.6
Fine_tuning.py	Entrenamiento y evaluación de modelos de extracción de relaciones entre entidades (RE).	5.7
Score_re.py	Evaluación de modelos de modelos de extracción de relaciones entre entidades (RE).	5.7
models_comparison.py	Compara los diferentes modelos RE.	5.7
Kfold_fine_tuning.py	Validación cruzada Kfolds para el modelo RE.	5.7
Sentences_extraction.py	Usando los módulos Segmentation.py y SegmentationHTML.py, aplica la fase de inferencia del modelo NER sobre la carpeta *DATA_TRAIN*.	5.8.1
Relation_extraction.py	Aplica la fase de inferencia del modelo RE sobre la carpeta *DATA_NER*.	5.8.2
Seccion_extraction.py	Proceso de post-procesamiento, sobre la carpeta *DATA_RE*.	5.9
Structure_extraction.py	Extracción de estructuras de celdas para complementar a Seccion_extraction.py, sobre la carpeta *DATA_RE*.	5.9
Seccion_extraction_score.py	Métricas complementarias de coherencia, para generar los devices.	5.9
EXCEL.py	limpiar y extraer la información del Database manual.	5.10
COMPARA_entities.py	Comparación a nivel de entidad.	5.10.1
COMPARA_devices.py	Comparación a nivel de dispositivo.	5.10.2

Apéndice J. Artículo derivado de la investigación

Datos bibliográficos

- **Titulo:** Named Entity Recognition for Performance and Synthesis information of Perovskite Solar Cells using Spacy
- **Congreso:** ICCSA 2025
- **Autores:** Mary Jiménez-Díaz, Alexander Sepúlveda-Sepúlveda, and Mónica Botero-Londoño
- **DOI:** `https://doi.org/10.1007/978-3-031-96997-3_19`
- **Estado:** Publicado
- **Abstract:**

  This paper presents a Named Entity Recognition (NER) approach for extracting relevant information related to the design and optimization of Perovskite solar cells (PSC). NER is a key component of Information Extraction (IE), allowing for structured knowledge retrieval from unstructured scientific texts. While NER has previously been applied to solar cell research, existing approaches often rely on computationally expensive models or large research teams. In contrast, we explore the use of SpaCy, a lightweight natural language processing framework, to develop an efficient and accessible model that can be implemented with limited computational resources.

  Our approach focuses on identifying key entities related to structural composition, synthesis methods, and device performance. To achieve this, we trained and evaluated our model using K-fold cross-validation, achieving an average precision of 89.94 %, recall of 92.47 %, and F1-score of 89.69 %.

  To assess the model's performance in a more realistic setting, we compared the extracted information to manual annotations in two reference databases: Odabasi (2019) and Jacobsson (2022). This evaluation focused on key entities related to perovskite solar cell layers, reinforcing the model's effectiveness in extracting relevant information.

  Our results highlight the potential of this methodology to facilitate knowledge extraction in PSC research, offering a cost-effective and accessible alternative to more complex machine learning models. Furthermore, this strategy can be extended to other scientific and technological fields requiring automated extraction of structured and unstructured text information.

Certificado de presentación



International Conference of Computational Science and Its Applications (ICCSA 2025)

Galatasaray University
Istanbul, Türkiye, June 30 - July 3, 2025

Certificate of Attendance

I hereby certify that
Mary Zuleika Jiménez Díaz
Universidad Industrial de Santander
Colombia

Registration type: Virtual

has attended virtually The 2025 International Conference on Computational Science and Its Applications (ICCSA 2025), which was held in a mixed form, with part of the delegates in presence and part online, co-organised by the Galatasaray University, June 30 - July 3, 2025

Presenting:

Named Entity Recognition for Performance and Synthesis information of Perovskite Solar Cells using Spacy

Osvaldo Gervasi
ICCSA 2025 General CoChair

Primera página del artículo

Named Entity Recognition for Performance and Synthesis Information of Perovskite Solar Cells Using SpaCy

Mary Zuleika Jiménez-Díaz(✉), Alexander Sepúlveda-Sepúlveda, and Mónica Botero-Londoño

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander, Bucaramanga, Santander, Colombia
mary2238319@correo.uis.edu.co, alexander.sepulveda@saber.uis.edu.co, mabotero@uis.edu.co

Abstract. This paper presents a Named Entity Recognition (NER) approach to extract key information related to the structural components, synthesis techniques, and photovoltaic performance metrics of Perovskite solar cells (PSCs). Since NER is a key component of Information Extraction (IE), which identifies and classifies key elements of a text, we propose the use of the *SpaCy* library to build an effective and accessible NER model, suitable for environments with limited computational capacity, unlike other previous works in this field, which make use of large-scale models or high computational resources. Our resulting model was evaluated using *K-fold cross-validation*, obtaining the mean scores of, precision of 89.94%, a recall of 92.47%, and an F1 score of 89.69%. To provide a test of the practical performance of the resulting model, implementing and comparing the obtained results with manual annotations in two Excel reference databases: Odabaşı (2019) [1] and Jacobsson (2022) [2], demonstrating the potential of our work to facilitate and accelerate knowledge extraction, and the possibility of extending this strategy to other scientific fields where automated text extraction is required.

Keywords: Natural language processing · NER · Perovskite Solar Cells · Spacy · Machine Learning · Named Entity Recognition

1 Introduction

In recent years, the development of renewable energy sources with low environmental impact has gained increasing attention, driven by global initiatives to limit global warming and mitigate the costs of traditional energy. According to the REN21 report [3], investment in renewable energy reached a historic high in 2023, reflecting its accelerated growth.

Perovskite solar cells (PSCs), classified as third-generation photovoltaic devices, have emerged as a strong contender to silicon-based technologies, primarily due to their high power conversion efficiency (PCE) and potential cost

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15649, pp. 303–319, 2025.
https://doi.org/10.1007/978-3-031-96997-3_19