HERRAMIENTA DE SOFTWARE PARA LA EXTRACCIÓN AUTOMÁTICA DE PARÁMETROS DE DESEMPEÑO A PARTIR DE PUBLICACIONES CIENTÍFICAS DE CELDAS SOLARES DE PEROVSKITA

Mary Zuleika Jimenéz Díaz

Trabajo de Grado para optar al título de Ingeniera Electrónica

Director

Franklin Alexander Sepúlveda Sepúlveda

Doctorado en Ingeniería Electrónica

Codirectores

Camilo Andrés Otálora Bastidas

Doctorado en ciencias Química

Cristian David Camacho Parra

Ingeniero Electricista

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2022

Dedicatoria

Este trabajo esta dedicado a todas aquellas personas que apoyaron el desarrollo y ejecución de esta tesis.

Agradecimientos

Agradezco a mi familia por el apoyo económico y moral que me brindaron durante el desarrollo de mi carrera.

Un reconocimiento y agradecimiento importante lo realizo a mi director, codirectores de trabajo de grado y miembros del grupo CEMOS y GISEL, por dedicar su tiempo, experiencia y conocimiento en la guía de mi proyecto.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Apéndices

pág.

Glosario

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).

V_{oc} (*Open-Circuit Voltage*) es el voltaje máximo disponible de una celda solar, y esto ocurre con corriente cero. El voltaje de circuito abierto corresponde a la cantidad de polarización directa en la celda solar debido a la polarización de la unión de la celda solar con la corriente generada por la luz (C.B.Honsberg and S.G.Bowden, 2019).

J_{sc} (*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).

PCE (*Power Conversion Efficiency*) es la relación entre la producción de energía de la celda solar y la energía de entrada del sol. Además de reflejar el rendimiento de la propia célula solar, la eficiencia depende del espectro y la intensidad de la luz solar incidente y de la temperatura de la célula solar (C.B.Honsberg and S.G.Bowden, 2019).

FF (*Fill Factor*) es un parámetro que, junto con Voc e Jsc , determina la potencia máxima de una célula solar. El FF se define como la relación entre la potencia máxima de la celda solar y el producto de Voc y Jsc (C.B.Honsberg and S.G.Bowden, 2019).

ETL (*Electron Transport Layer*) es la capa de la celda solar de Perovskita (PSC) que se encarga de ser la transportadora de electrones.

HTL (*Hole Transporting Layer*) es la capa de la celda solar de Perovskita (PSC) que se encarga de ser la transportadora de huecos.

Tubería (*Pipeline*)es un término inglés que puede traducirse como "*tubería*". En el entorno de la informática se refiere a el 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 sofware (Porto and Merino, 2015).

Lematización (*Lemmatization*) es el proceso de reducir una palabra a su raíz o unidad base, normalmente previo a el desarrollo del procesamiento de datos. Implica el uso de su forma de diccionario o a 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).

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 modulo "re" (Python Software Foundation, 2021).

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).

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).

Clusterización (*Clustering*) es una tarea que tiene como finalidad principal lograr el agrupamiento de conjuntos de objetos no etiquetados, para lograr construir subconjuntos de datos conocidos como Clusters. Cada cluster dentro de un grafo está formado por una colección de objetos o datos que a términos de análisis resultan similares entre si, pero que poseen elementos diferenciales con respecto a otros objetos pertenecientes al conjunto de datos y que pueden conformar un cluster independiente (GraphEverywhere, 2022).

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).

Resumen

Título: HERRAMIENTA DE SOFTWARE PARA LA EXTRACCIÓN AUTOMÁTICA DE PARÁMETROS DE DESEMPEÑO A PARTIR DE PUBLICACIONES CIENTÍFICAS DE CELDAS SOLARES DE PEROVSKITA [*]

Autor: Mary Zuleika Jiménez Díaz [**]

Palabras Clave: Minería de datos, Procesamiento del lenguaje natural, Perovskitas, Celdas solares.

Descripción: El creciente interés en la tecnología fotovoltaica basada en Perovskitas ha llevado a un aumento significativo del número de artículos científicos publicados, lo que implica un aumento en la cantidad de información disponible para analizar. Dada la diversidad y gran cantidad de fuentes, los análisis profundos de la literatura relacionada, ahora toman gran tiempo.

En el presente trabajo se plantea una solución para agilizar el análisis y recolección de información, mediante aplicaciones relacionadas con el aprendizaje automático, usando herramientas de procesamiento de lenguaje natural (NLP) y minería de texto. Para esto se desarrolla una herramienta software de aprendizaje automático en lenguaje Python, que permite la extracción automática de capas de la celda de Perovskita como la ETL, la HTL y la *Perovskita*, y de parámetros de desempeño de la celda Perovskita como el V_{oc}, el J_{sc}, el PCE y el FF, basada en la librería Python *spaCy* 3.0 a partir de publicaciones de estudios científicos en formato PDF almacenados en una base de datos.

La herramienta de software cuenta con la funcionalidad de comandos de linea y se encuentra guardada en un repositorio virtual de Github. Además se adicionan dos trabajos a futuro, una interfaz gráfica desarrollada en Python para la corrección de datos de entrenamiento y un acercamiento a la minera de texto aplicada a tablas contenidas en PDFs.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Franklin Alexander. Codirector 1: Camilo Andrés Otálora Bastidas. Codirector 2: Cristian David Camacho Parra.

Abstract

Title: SOFTWARE FOR AUTOMATIC EXTRACTION OF PERFORMANCE PARAMETERS FROM SCIENTIFIC PUBLICATIONS ON PEROVSKITE SOLAR CELLS [*]

Author: Mary Zuleika Jiménez Díaz [**]

Keywords: Data mining, Natural language processing, Perovskites, Solar cells.

Description: The growing interest in Perovskite-based photovoltaic technology has led to a significant increase in the number of scientific papers published, which implies an increase in the amount of information available for analysis. Given the diversity and large number of sources, in-depth analyses of the related literature now take a great deal of time.

This paper proposes a solution to speed up the analysis and collection of information through applications related to machine learning, using natural language processing (NLP) and text mining tools. For this purpose, a machine learning software tool is developed in Python language, which allows the automatic extraction of Perovskite cell layers such as ETL, HTL and *Perovskite*, and Perovskite cell performance parameters such as V_{oc}, J_{sc}, PCE and FF, based on the Python library *spaCy* 3.0 from publications of scientific studies in PDF format stored in a database.

The software tool has line command functionality and is stored in a virtual repository on Github. In addition, two future works are added, a graphical interface developed in Python for the correction of training data and an approach to text mining applied to tables contained in PDFs.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Franklin Alexander. Codirector 1: Camilo Andrés Otálora Bastidas. Codirector 2: Cristian David Camacho Parra.

Introducción

En el contexto actual, el desarrollo energético se ha alineado hacia las energías renovables de bajo impacto ambiental, las cuales son necesarias para cumplir los objetivos globales de mantener el aumento de la temperatura global inferior a 2 °C. Según el reporte global del REN21, la energía renovable tuvo otro año de aumento récord en 2020, durante el cual la capacidad de energía renovable instalada aumento en más de 256 Gigavatios [GW], su mayor aumento hasta la fecha (como se observa en la Figura 1). Las adiciones de $256,6[GW]$ consistieron en $139,4[GW]$ de energía solar fotovoltaica, $93,0[GW]$ de energía eólica, $19,4[GW]$ de energía hidroeléctrica, $4,6[GW]$ de bioenergía, $0,1[GW]$ de energía geotérmica, $0,1[GW]$ de CSP y $\sim 0[GW]$ de energía oceánica. A nivel mundial, aproximadamente el $29\,\%$ de nuestra electricidad proviene ahora de energía renovable, y esta sigue creciendo (S. REN21, 2021b).

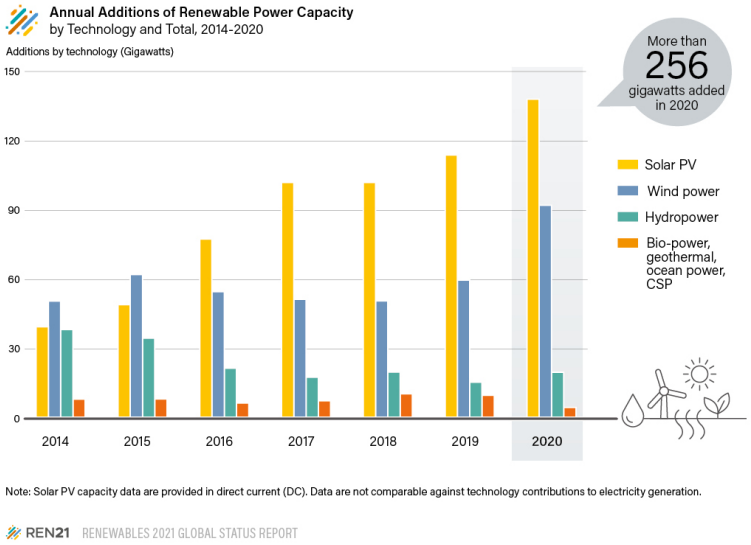

Figura 1. Capacidad de potencia adicional anual de energía renovable, tomada de la REN21 (S. REN21, 2021b).

La energía solar fotovoltaica, que convierte la radiación solar en energía eléctrica mediante procesos físico-químicos, es una de las energías renovables más populares y es considerada una de las más limpias. En particular, en el año 2020, a pesar de la pandemia causada por el COVID-19, la energía solar fotovoltaica obtuvo un récord en la capacidad instalada (dentro y fuera de la red), alcanzando un estimado de 760 GW como se ve en la Figura 2 (S. REN21, 2021a).

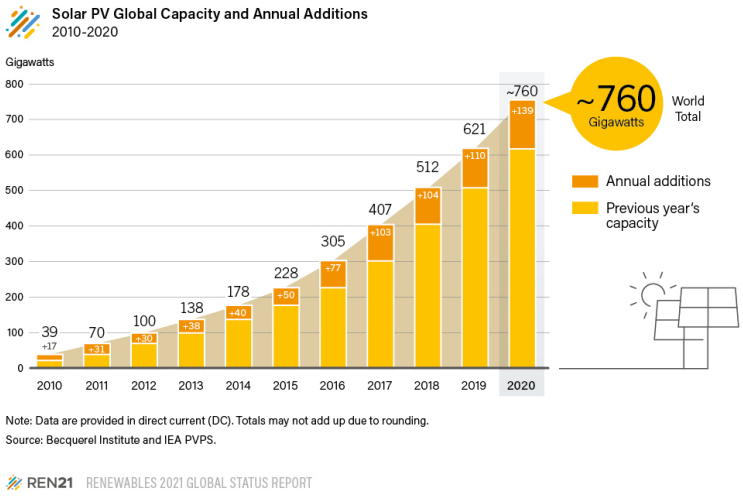

Figura 2. Capacidad global de energía solar fotovoltaica y adiciones anuales de los años 2010-2020, tomada de la REN21 (S. REN21, 2021b).

Debido al creciente volumen de energía generado se hace necesario incrementar la eficiencia y disminuir costos en la generación de la misma. Con el fin de mejorar la eficiencia y los costos, recientemente se ha planteado el uso de celdas solares de tercera generación como alternativa a las celdas convencionales de primera y segunda generación. Dentro de estas celdas de tercera generación se encuentran las conocidas como celdas solares de Perovskita, que tiene bajos costos de fabricación y alta eficiencia (Graetzel et al., 2012). Las celdas de perovskita (PSC, *Perovskite Solar Cells*), son celdas con costos competitivos y producen un impacto ambiental menor que las celdas

solares convencionales de silicio (Vidal et al., 2021). En consecuencia, un problema relevante consiste en mejorar la capacidad de la celda para convertir los fotones provenientes de la luz solar en electrones que puedan ser recolectados y encausados para así generar una corriente eléctrica.

Sin embargo, este proceso de generar nuevos materiales de Perovskitas con mayor desempeño se ha venido desarrollando de manera empírica, con sus consecuentes costos en recursos y mano de obra. En tal sentido, utilizar la experiencia consignada, aunque de manera dispersa, en los artículos de la literatura científica sería una forma de mejorar el proceso de diseño de celdas solares de tercera generación mediante la reducción de los tiempos de desarrollo y de los costos de diseño de un nuevo material.

Debido a las ventajas en costo y eficiencia, el interés por parte de investigadores en las PSCs se ha incrementado notablemente, provocando un incremento relevante en la cantidad de resultados reportados en forma de artículos científicos, lo que a su vez genera un aumento considerable de información útil que puedan alimentar los métodos de descubrimiento del conocimiento. Ello a su vez aumenta las posibilidades para el descubrimiento y diseño de nuevos materiales con propiedades mejoradas a través técnicas de búsqueda y análisis de información sobre datos mediante el aprendizaje automático (*Machine Learning*) (Çağla Odabaşı and Yıldırım, 2019; Odabasi and Yildirim, 2020; Parikh et al., 2022; Li et al., 2019). Sin embargo, la gran cantidad de información y su constante actualización hacen que la recolección y clasificación de los datos relevantes sea una tarea ardua e intensiva en tiempo y costos por parte de mano de obra calificada. Es aquí donde entraría el uso de técnicas de *Minería de Texto* que pudieran facilitar la extracción automática y rápida de la información relevante contenida en artículos científicos.

Una parte muy importante de la Minería de Texto es la extracción automática de la información relevante (IE, *Information Extraction*) referente a las configuraciones y desempeño de celdas solares contenidas en los artículos científicos; para lo cual, a groso modo, se requiere la solución de dos problemas: primero, el reconocimiento de entidades nombradas (NER, *Named Entity Recognition*); y el segundo, detección de relación entre las entidades detectadas (RE, *Relation Extraction*). El sistema toma como entrada las cadenas de texto de los archivos en formato PDFs, que posteriormente son convertidos a archivos de texto en formato plano. Luego, esa gran cadena de texto es segmentada en oraciones y se realiza el proceso de reconocimiento de entidades(Olivetti et al., 2020). Finalmente, los datos de las configuraciones han de obtenerse mediante la aplicación de un algoritmo de detección de relaciones entre entidades. El desarrollo de estas tareas se realiza haciendo uso de librerías de procesamiento de lenguaje natural (NLP, *Natural Language Processing*) tales como *SpaCy* (Honnibal and Montani, 2015), NLTK y *FreeLing* (Padró and Stanilovsky, 2012), entre otras.

El presente trabajo se enfoca en el proceso NER, y no incluye el proceso de detección de relación entre entidades. Aunque existen herramientas de reconocimiento de entidades de materiales de índole química disponibles en la red, tales como *ChemDataExtractor* (Swain and Cole, 2016), *ChemicalTagger* (Hawizy et al., 2011) y *OSCAR4* (Jessop et al., 2011), estas se enfocan en la extracción de entidades científicas correspondientes a elementos y/o compuestos químicos. Las herramientas antes mencionadas no están optimizadas para el objeto de estudio en particular al que atañe el presente trabajo, y además no permiten fácilmente el entrenamiento de nuevas entidades.

Con el desarrollo del presente trabajo de grado se busca realizar una herramienta automá-

tica para el reconocimiento de entidades correspondientes a parámetros de desempeño de celdas solares de Perovskitas: V_{oc} (tensión de circuito abierto), J_{sc} (corriente de circuito cerrado), *PCE* (Eficiciencia de Conversión de Potencia) y FF (Fill Factor). Adicionalmente, se reconocen las entidades asociadas a la composición de las capas principales (HTL, ETL y Perovskita). Para este propósito, en el presente trabajo se hace uso de la librería *spaCy* 3.0 para la tarea de NLP, y para extraer el texto se utiliza *PyMuPDF*.

Los capítulos dos y tres se centran en el desarrollo de la contextualización de los temas y métodos relevantes para el proyecto. En el capitulo cuatro se describe el método completo escogido, conformado por tres partes principales: El preprocesamiento de datos, el entrenamiento del modelo y la aplicación del modelo. El quinto capítulo se centra en la aplicación del método del cuarto capítulo y el análisis de los resultados, en el caso de la creación y aplicación de un modelo NER para extraer las capas y parámetros de desempeño de artículos científicos de celdas solares de Perovskita en formato PDF. Luego los capítulos sexto, séptimo y octavo, presentan las recomendaciones, los trabajos futuros y las conclusiones, respectivamente. Además se incluyen cinco apéndices en donde se amplia la información del proyecto.

1. Objetivos

1.1. Objetivo general

Desarrollar una herramienta de software para la extracción automática de parámetros de desempeño, basado en la librería spaCy de Python, a partir de publicaciones científicas de celdas solares de Perovskita.

1.2. Objetivos específicos

- Crear y aplicar un módulo en Python de limpieza y decodificación de texto para estandarizar los datos extraídos.

- Desarrollar un modelo de procesamiento de lenguaje natural (NLP) haciendo uso de la librería Spacy, para la extracción automática de los parámetros de desempeño: Voltaje de circuito abierto (V_{oc}), Corriente de corto circuito (J_{sc}), eficiencia de conversión de potencia (PCE) y factor de llenado (FF).

- Realizar la evaluación del modelo NLP usando un porcentaje de las publicaciones contenidas en la base de datos.

2. Celdas Solares de Perovskita

La *Perovskita*, como compuesto a base de titanato de calcio ($CaTiO_3$), fue descubierta por el geólogo alemán Gustav Rose en 1839 y toma su nombre en honor al mineralogista ruso Lev Perovski; sin embargo, en la actualidad, se considera Perovskita a todo aquel compuesto con la misma estructura cristalina que el titanato de calcio (Rojas and Córdova, 2021). Aunque las primeras investigaciones con la perovskita como absorbente solar se dieron a principios de 1990 (Tonui et al., 2018), fué tan solo hasta 2009 que tuvo su primera demostración (Kojima et al., 2009).

2.1. Composición y propiedades

En términos generales, los compuestos denominados como Perovskitas cumplen con la formula ABX_3 (ver figura 3), donde A es un catión orgánico (e.g. Metil-amonio $CH_3NH_3^+$), B es un catión metálico (e.g. Pb_2^+) y X representa el haluro anión (por ejemplo, I^-) (Shi and Jayatissa, 2018). Las celdas solares de Perovskita se conocen con ese nombre ya que emplean como semiconductor absorbente de la radiación solar un compuesto con estructura tipo Perovskita.

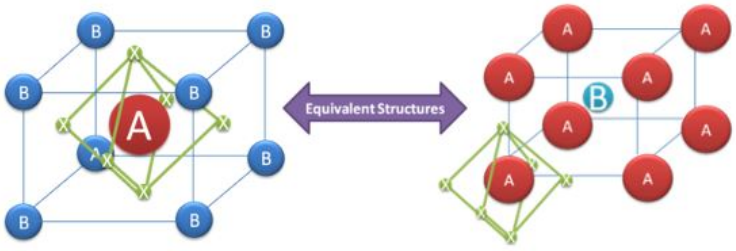

Figura 3. Estructura cristalina de la Perovskita genérica de la forma ABX_3. La estructura de la izquierda se dibuja de manera que el átomo B está en la posición (0,0,0), mientras que la estructura de la derecha está dibujada de manera que el átomo A está en la posición (0,0,0). Tomada de DS New Energy (2019).

Los principios de la unión *PN* utilizados para describir las celdas solares basadas en silicio siguen siendo aplicables para caracterizar las propiedades de las celdas solares de perovskita (Kim et al., 2018). Normalmente, las celdas solares de Perovskita están constituidas de varias capas uniformes y lineales (ver figura 4). En sus extremos tiene dos capas, la transportadora de electrones (ETL, *Electron Transport Layer*) y la transportadora de huecos (HTL, *Hole Transport Layer*). En cada uno de dos bordes se localiza un electrodo conductor metálico y como capa central esta la Perovskita.

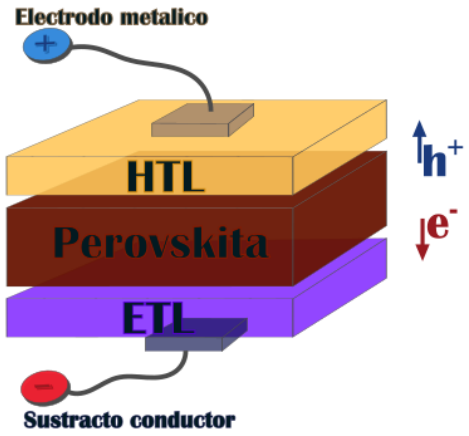

Figura 4. Representación gráfica de la estructura y las capas contenidas en una celda solar de Perovskita.

Las celdas solares que usan la Perovskita como la capa absorbente de luz tiene en comparación con otras celdas solares, relativa alta eficiencia, bajo costo de fabricación y parámetros de caracterización como el *PCE* (eficiencia de conversión de Potencia), el V_{OC} (Voltaje de circuito abierto), la J_{SC} (Corriente de corto circuito) y el FF (factor de llenado). Además las celdas solares de Perovskita gozan de características como (Ross, 2017):

- Elevados *coeficientes de extinción*; es decir, es un compuesto muy eficiente para la absorción de la luz a una determinada longitud de onda.

- Amplio *espectro de absorbancia*, hasta cerca del infrarrojo, lo que le da la capacidad de absorber la mayor parte del espectro visible.

- Elevada *movilidad intrínseca* de cargas, es decir el movimiento entre las capas de la banda conducción o por los huecos en la banda de valencia al incidir la luz.

- Elevada *polaridad*, que hace que se tenga un gran separación entre las cargas eléctricas dentro las moléculas.

2.2. La ciencia computacional de los materiales

En el campo del descubrimiento de nuevos materiales mediante ciencia computacional, los datos son los recursos de insumo y el conocimiento es el objetivo a conseguir mediante estos procedimientos. 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 basado en el razonamiento humano tradicional. En tal sentido, la combinación de conjuntos de datos, el aprendizaje automático (*Machine Learning*) y métodos de cómputo de alto rendimiento resultan de notable ayuda en el área de investigación de materiales (Himanen et al., 2019).

En cuanto a los datos, es importante tener en cuenta que las Perovskitas han tenido un gran ascenso entre las celdas solares, las cuales han acaparado la atención de la comunidad científica en el campo de la fotovoltaica en los últimos nueve años, como se observa en la Figura 5. A los 4

años de su estabilización como tecnología ya habían igualado las eficiencias de las celdas solares de telururo de cadmio (CdTe), que ha estado durante más de 40 años y partir de 2018 las celdas solares de Perovskita ha empezado a superar a otras tecnologías (DS New Energy, 2019).

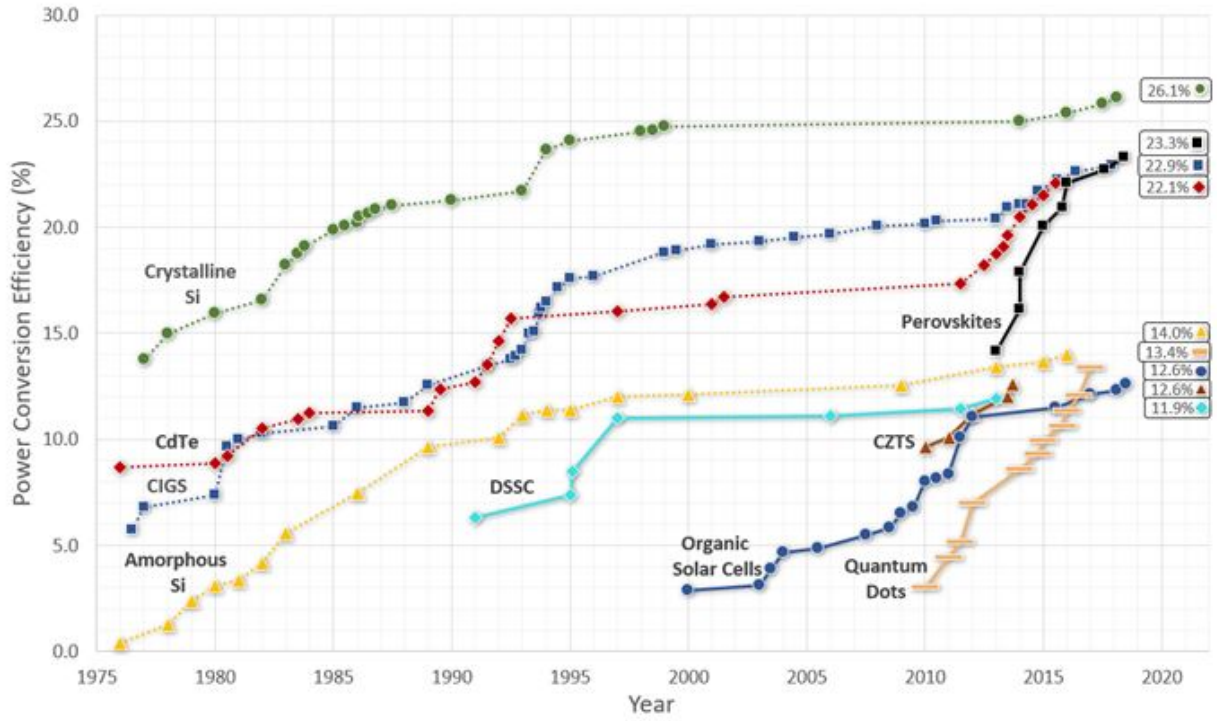

Figura 5. Comparación entre el crecimiento de tecnologías de celdas solares respecto a su PCE, tomada de DS New Energy (2019). La información actualizada de la gráfica se encuentra en el NREL (2022).

Los datos encontrados en las bases de datos de *Scopus*, *ScienceDirect* y *Web of Science* (ver figura 6 y 7) con la cantidad de publicaciones por año sobre las celdas solares de Perovskita, se muestra un creciente interés de la comunidad científica en el desarrollo de esta tecnología a medida que pasan los años.

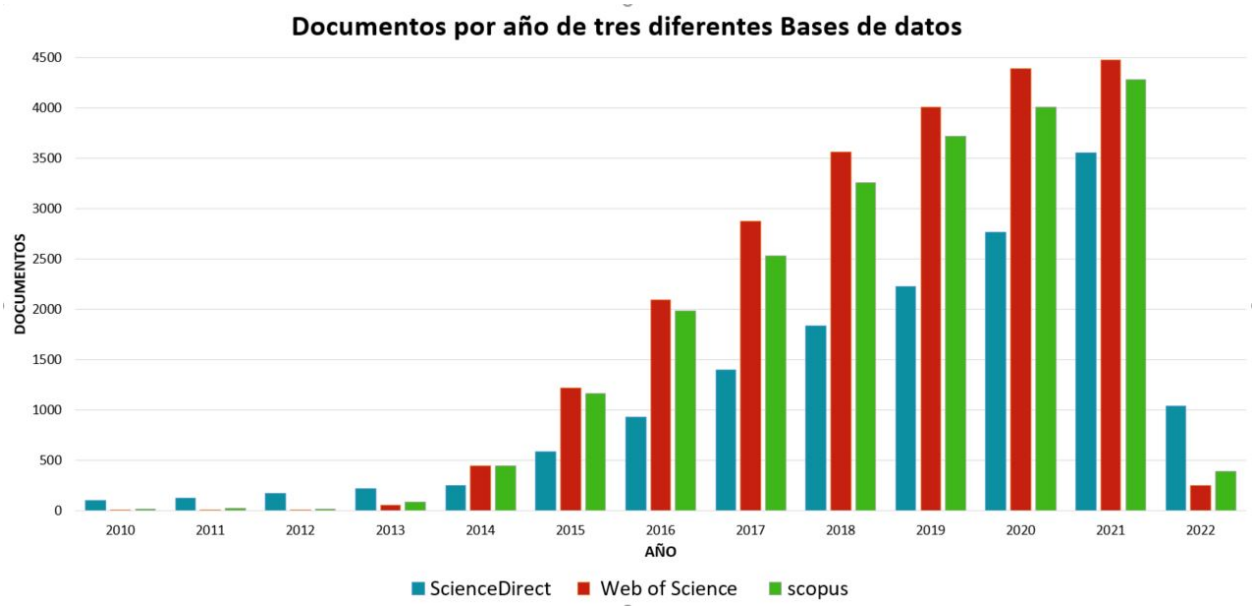

Figura 6. Gráfica con los valores de documentos sobre celdas solares de Perovskita según la base de datos que se analiza cada año, revisado durante enero 2022.

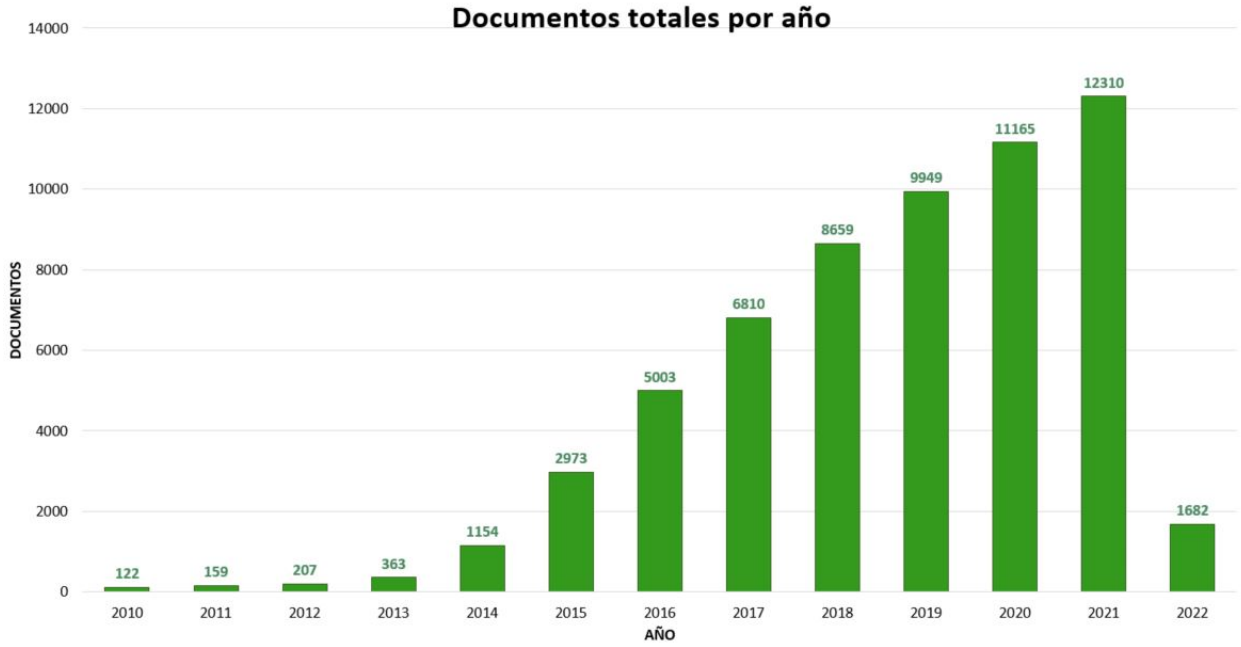

Figura 7. Gráfica con los valores de documentos sobre celdas solares de Perovskita totales cada año, revisado durante enero 2022.

3. Procesamiento de Lenguaje Natural para Minería de Texto

El NLP actualmente corresponde a una subárea de la inteligencia artificial (Fig. 8) que se centra en la interpretación automática del lenguaje humano; es decir, se enfoca en procesar la información comunicada y no simplemente analiza las secuencias de letras y sonidos presentes en los procesos de la comunicación humana. En particular, es una herramienta invaluable en el análisis, la agregación y la simplificación de texto (Kang et al., 2020). Originalmente no fue concebido como una rama de la inteligencia computacional; sin embargo, las características de lo problemas a resolver hicieron que la mayoría de los sus sistemas orientados al análisis de textos basen ahora su funcionamiento en la utilización de técnicas de aprendizaje automáticos, en lugar de utilizar sistemas basados en reglas particulares de la misma aplicación (Alias and Cassanelli, 2019).

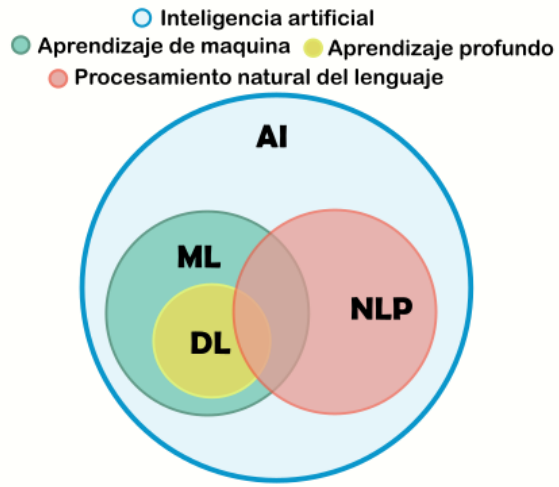

Figura 8. Diagrama sobre la distribución general del procesamiento natural del lenguaje como parte de la inteligencia artificial. Digrama modificado de la figura encontrada en AthenaTech LLC (2019).

El NLP permite aplicar una gran gama de tareas relacionadas con el lenguaje natural en

todos los niveles, como el análisis sintáctico, el etiquetado de partes del discurso (POS), el reconocimiento de entidades nombradas (NER), la traducción automática y hasta sistemas completos de dialogo. En particular, el presente trabajo atañe con la temática NER, que es una tarea que consiste en analizar textos completos a fin de localizar y clasificar las entidades nombradas en categorías predefinidas tales como lugares, organizaciones, entre otras, y en el presente caso, en tipos de medidas de eficiencia de celdas solares y capas de la Perovskita.

3.1. Redes Neuronales

Las redes neuronales son una de las técnicas más ampliamente usadas para el desarrollo de herramientas de NLP. Su creación se basa al tratar de realizar una analogía con el comportamiento de las neuronas del cerebro humano, el cual esta compuesto de una enorme cantidad de neuronas biológicas con bajas capacidades de procesamiento, pero que al interconectarlas pueden realizar tareas alto grado de complejidad (Salas, 2004).

El procesador elemental llamado neurona posee la capacidad limitada de calcular, generalmente una suma ponderada de sus entradas y luego aplicar una función de activación para obtener una señal que será transmitida a la próxima neurona. Estas neuronas artificiales se agrupan en capas o niveles y poseen un alto grado de conectividad entre ellas, conectividad que es ponderada por los pesos (Salas, 2004). Las redes neuronales se caracterizan principalmente por: (Izaurieta and Saavedra, 2000)

- Tener una inclinación natural a adquirir conocimiento a través de la experiencia, que se almacena a igual que en un cerebro con un peso relativo de las conexiones inter-neuronales.

- Tiene una altísima plasticidad y gran adaptabilidad, siendo capases de cambiar dinámicamente.
- Poseen un alto nivel de tolerancia a las fallas, es decir pueden sufrir daños considerables y aún así continuar con un buen comportamiento.
- Tienen un comportamiento altamente no-lineal, permitiéndoles procesar información de otros fenómenos no-lineales.

3.2. Herramientas de NLP

Los avances generados gracias al uso de herramientas NLP han provocado cambios radicales en las prácticas comerciales, que han 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*), entre otros (Kang et al., 2020). 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.

A grandes rasgos, para el desarrollo de un sistema basado en NLP se requieren 3 fases (ver figura 9): pre-procesamiento de texto, representación de texto, y finalmente el entrenamiento de modelo y aplicación del modelo mediante algoritmos apropiados para la tarea en particular. El pre-procesamiento limpia el texto para así mejorar la eficiencia y precisión de las de las etapas de estandarización gramatical, verificación de errores ortográficos, tokenización, etiquetado gramatical (POS), la eliminación de palabras vacías y la lematización. Después de preprocesar el texto,

se debe elegir una manera de representar las palabras en la computadora y llevar las palabras y secuencias de palabras a vectores y/o matrices. Una vez se tienen estos vectores de palabras es posible aplicar alguno de los algoritmos usados en NLP, cuyos modelos se deben entrenar, con el fin de resolver el problema en cuestión (Kang et al., 2020).

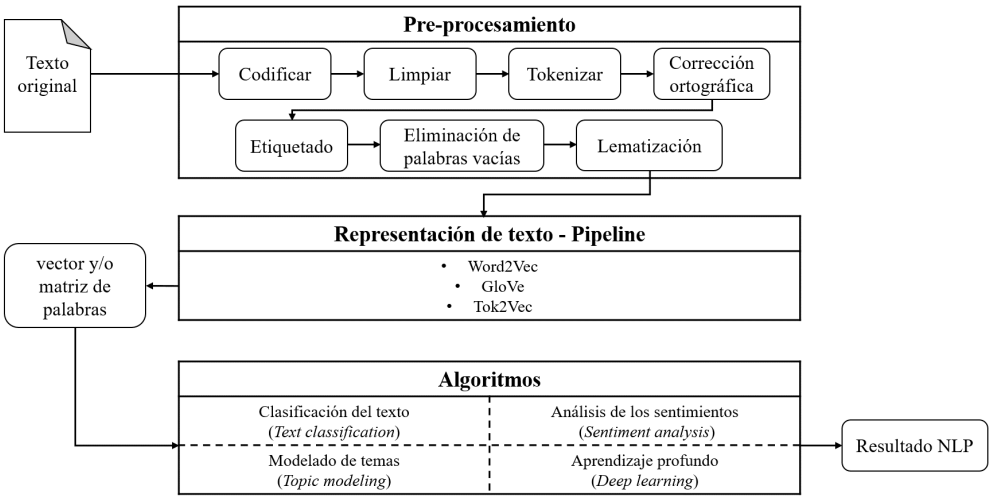

Figura 9. Diagrama de flujo que describe las partes estándar que componen a una herramienta de NLP.

Entre las herramientas en forma de librerías disponibles para uso y modificación científico de modelos NLP se mencionan: *NLTK* (Loper and Bird, 2002), *spaCy 3.0* (Honnibal and Montani, 2015), *CoreNLP* (Manning et al., 2014), *AllenNLP* (Gardner et al., 2017), *Vertex AI* (Google Cloud, 2021). En la tabla 1 se muestran estas herramientas disponibles y se hace una revisión corta de las mismas.

Tabla 1
Resumen de algunos de las herramientas de NLP que permiten el entrenamiento o generación de modelos de extracción de entidades nombradas (NER).

Nombre	***Sitio Web***	***Resumen***
NLTK	`http://nltk.sf.net/`	Es una plataforma diseñada inicialmente como material didáctico y luego convertido en un proyecto de código abierto para crear programas Python que funcionen con datos de lenguaje humano. Proporciona interfaces y recursos léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto. Desarrollado por el departamento de computación y ciencia de la información de la la Universidad de Pennsylvania (Loper and Bird, 2002).
spaCy	`https://spacy.io`	Es una biblioteca gratuita de código abierto para el procesamiento avanzado del lenguaje natural (NLP) en Python. Permite la modificación, actualización o generación de modelos de redes neuronales o sistemas de extracción de información o de comprensión del lenguaje natural. Diseñado específicamente para la producción de aplicaciones que procesan y "comprenden" grandes volúmenes de texto (Honnibal and Montani, 2015).
CoreNLP	`https://stanfordnlp.github.io/CoreNLP/`	Es un pipeline extensible que proporciona análisis básico del lenguaje natural (NLP) en Java con una sola API. Desarrollado por la universidad de Stanford para la academia y luego lanzado con una licencia de código abierto (Manning et al., 2014).
AllenNLP	`https://allennlp.org/`	Es una plataforma para la investigación en profundidad de métodos de aprendizaje en el procesamiento del lenguaje natural (NLP) y de aprendizaje profundo. Construido sobre sobre PyTorch (Paszke et al., 2017) y mantenido por el Allen Institute for AI, en estrecha colaboración con investigadores de la Universidad de Washington y otros lugares (Gardner et al., 2017).
Vertex AI	`https://cloud.google.com/vertex-ai`	Es una plataforma administrada de aprendizaje automático que le brinda todos los servicios en la nube de Google en un solo lugar para implementar y mantener modelos de inteligencia artificial. Con la opción que elijas para el entrenamiento, puedes guardar modelos, implementar modelos y solicitar predicciones con Vertex AI (Google Cloud, 2021).

En el presente trabajo se opta por usar *spaCy 3.0* por ser una de las herramientas de código libre más sencillas, posee una curva de aprendizaje más rápida, y además permite una gran variedad de configuraciones. Esta librería esta basada en *Cython* y permite el entrenamiento de modelos de redes neuronales. El hecho de usar *Cython* le permite ofrecer una API sencilla en Python con un rendimiento similar a del lenguaje C. Las características y capacidades de *spaCy 3.0* se muestran en la Tabla 2.

Tabla 2
Tabla de características y capacidades de la librería spaCy 3.0, tomada de la pagina oficial Honnibal and Montani (2015).

NOMBRE	***DESCRIPCIÓN***
Tokenización	Segmentar el texto en palabras, signos de puntuación, etc.
Etiquetado de parte de voz (POS)	Asignar tipos de palabras a tokens, como verbo o sustantivo.
Análisis de dependencia	Asignar etiquetas de dependencia sintáctica, describiendo las relaciones entre tokens individuales, como sujeto u objeto.
Lematización	Asignar las formas base de las palabras. Por ejemplo, el lema de *era* es *ser* y el lema de *ratas* es *rata*.
Detección de límites de oración (SBD)	Encontrar y segmentar frases individuales.
Reconocimiento de entidad nombrada (NER)	Etiquetado de objetos denominados *del mundo real*, como personas, empresas o ubicaciones.
Vinculación de entidades (EL)	Desambiguando entidades textuales a identificadores únicos en una base de conocimiento.
Semejanza	Comparar palabras, tramos de texto y documentos y qué tan similares son entre sí.
Clasificación de texto	Asignar categorías o etiquetas a un documento completo o partes de un documento.
Coincidencia basada en reglas	Encontrar secuencias de tokens basadas en sus textos y anotaciones lingüísticas, similares a las expresiones regulares.
Capacitación	Actualizar y mejorar las predicciones de un modelo estadístico.
Serialización	Guardar objetos en archivos o cadenas de bytes.

Los modelos *NER* de *spaCy 3.0* pueden ser generados a partir de modelo globales estándar o pipelines entrenados del idioma a trabajar e ir así agregando o entrenando nuevas características, modificando lo necesario para amoldar la aplicación a las necesidades del nuevo modelo. Los modelos globales o pipelines entrenados que ofrece *spaCy 3.0* generalmente contienen un etiquetador, un lematizador, un analizador, un reconocedor de entidades nombradas (Honnibal and Montani, 2021d). Además posee:

- *Binary weights* o pesos binarios para los etiquetadores del discurso, los analizadores de dependencias y reconocedores de entidades nombradas (NER).
- *Lexical entries* o grupos de palabras con sus atributos independientes de contexto.
- *Data files*, es archivos de datos con reglas de lematización y tablas de búsqueda.
- *Word vectors*, representaciones del significado de palabras que permiten determinar qué tan similares son entre sí.

3.3. Modelos NER usando Spacy

SpaCy 3.0 ofrece una variedad de componentes que pueden ser aprovechadas para la obtención de un modelo NER aplicable a la detección de entidades sobre artículos científicos del área de celdas solares de Perovskita. Se mencionan alguna de ellas.

3.3.1. Tokenización. La tokenización para los modelos NER es la tarea de dividir un texto en segmentos significativos, llamados tokens, los cuales son unidades que se consideran inseparables para ayudar al reconocimiento de las entidades del modelo (Pai, 2020). La librería

spaCy 3.0 separa de forma automática la puntuación que no forma parte integral de una palabra, así como a las comillas, comas y puntación final como un token. Por ejemplo, la frase "*Before the passivation,the cell had a J_{sc} of 18.4 mA/cm2, V_{oc} of 0.90 V, FF of 0.68, and PCE of 11.3%.*", al ser tokenizada por la librería se dividirá como se ve en la Figura 10.

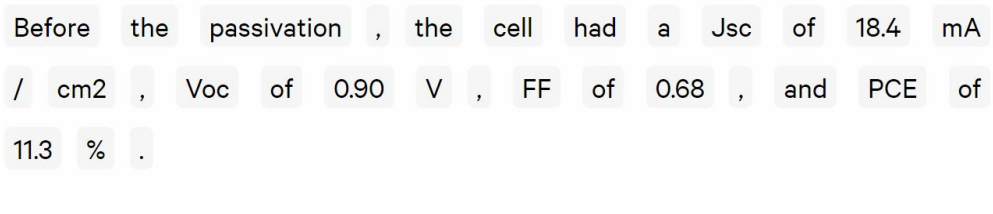

Figura 10. Representación gráfica de la tokenización realizada por la librería spaCy 3.0 para textos en ingles obtenida con el visualizador demo "*Rule-based Matcher Explorer*" (Honnibal and Montani, 2021c).

3.3.2. Entidades predeterminadas. Las entidades son definidas como nombres propios, elementos, compuestos, figuras públicas, puntos de referencia, etc, palabras claves sobre un tema en un texto, ejemplo Figura 11.

Aunque la librería *spaCy 3.0* tiene una serie de entidades nombradas pre-entrenadas, esta permite la definición de nuevas entidades nombradas para modelos usando dos modos 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.

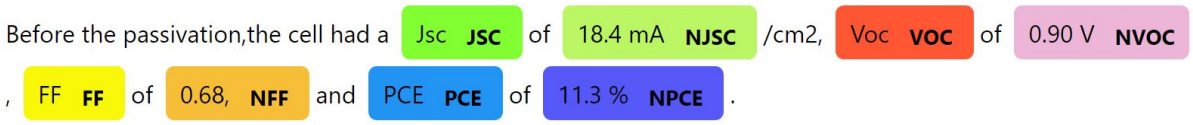

Figura 11. Ejemplo de entidades nombradas encontradas en un texto usando un modelo entrenado en spaCy 3.0.

La libreria *spaCy 3.0* nos da recomendaciones al escoger y trabajar las entidades nombradas:

- Deben de tener los nombres lo más sencillos y genéricos posible.
- Asegurarse que cada token sea parte de una sola entidad.
- Tener cuidado de una entidad no sea mas pequeña que un token, es decir no cortar o separar la unidad del token al obtener la entidad.

3.3.3. Entrenamiento. *spaCy 3.0* se puede adecuar para que la actualización de los pesos de la red neuronal la realice por muestra o por lotes (batches). El modelo predice las entidades para un grupo de ejemplos usando los parámetros de la iteración actual, luego compara las predicciones con las respuestas de referencia para así decidir como cambiar los parámetros del modelo NER. Este proceso se repite hasta cumplir la condición de parada pre-definida para el proceso (Honnibal and Montani, 2021a). La Figura 12 corresponde a una representación gráfica de este proceso.

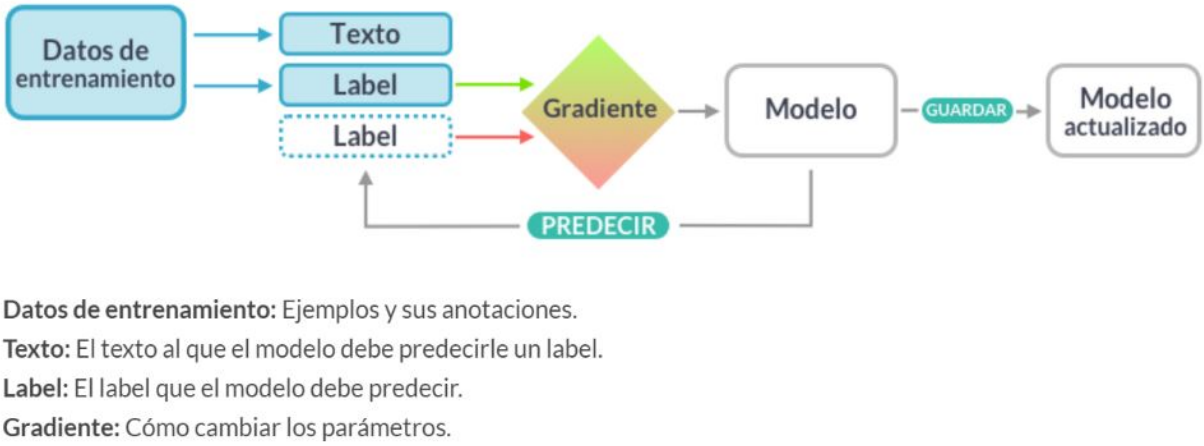

Figura 12. Diagrama de bloques del ciclo de entrenamiento ejecutado por la librería spaCy 3.0, tomado de las imágenes del curso online "NLP AVANZADO con spaCy"(Honnibal and Montani, 2021a).

3.3.4. Regularización. *spaCy 3.0* ofrece la posibilidad de regularización mediante la ***tasa de olvido (Dropout)*** (Antona Castañares, 2020). La idea general es la de 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 (Baciero Fernández, 2020). Con la librería *spaCy 3.0* se aplica usando la función "*drop*" del método "*nlp.update*", que tiene como valor por defecto el valor 0.1 siendo 1 el máximo aplicable.

Adicionalmente, se tiene la posibilidad de aplicar regularización por ***Parada temprana*** (Gencay and Qi, 2001), donde se recomienda realizar pruebas de entrenamiento para encontrar el numero de iteraciones mas óptimo para el modelo, evitando en la medida de lo posible que se sobre actualicen los pesos del modelo y empeore el entrenamiento. Con la librería *spaCy 3.0* se puede realizar observando durante las iteraciones la función de perdidas "*losses*" del método "*nlp.update*" hasta estabilice su valor o cuando el error de la fase de pruebas aumente en vez de

disminuir, tomando ese numero de iteraciones como el que debe ser usado para el entrenamiento.

3.3.5. Evaluación de modelos. *spaCy 3.0* cuenta con los métodos "*scores*" y "*evaluate*". Estos métodos para la evaluación de modelos NER, primero toman oraciones o conjuntos de datos con sus entidades nombradas verdaderas, pertenecientes a los datos de entrenamiento y/o datos de evaluación, les aplica el modelo entrenado y les extrae la entidades nombradas resultantes. Luego aplican las métricas de análisis estadístico usando las entidades nombradas verdaderas y las entidades nombradas resultantes. Las métricas del análisis estadístico de la evaluación de modelos NER son (Murphy, 2012)(Eisenstein, 2018), Figura 13:

- **Precisión**, permite medir la calidad del modelo en tareas de clasificación. Es la relación entre la verdaderas entidades nombradas (true positive) y las entidades nombradas extraídas por el modelo (elementos seleccionados).
- **Recall**, es la métrica de exhaustividad, es decir es la métrica que dice cuantas identidades verdaderas (true positive) el modelo logra identificar.
- **F-score**, puede resumirse como la media armónica entre la precisión y el recall.

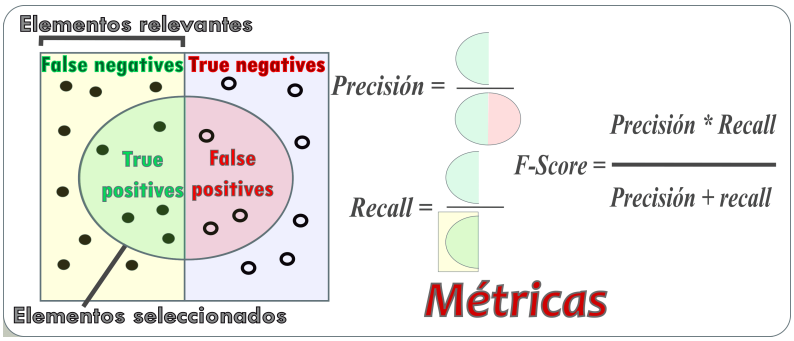

Figura 13. Resumen de las métricas de evaluación.

Otra forma de evaluar modelos NER consiste en utilizar métodos de ***re-muestreo*** (resampling) que permiten estimar la capacidad predictiva de los modelos, cuando se aplican a nuevas observaciones, ajustando el modelo al obtener un subconjunto de observaciones del conjunto de entrenamiento y evaluarlo con las observaciones restantes. Este proceso se repite múltiples veces y los resultados se agregan y promedian. Gracias a las repeticiones, se compensan las posibles variaciones que puedan surgir por el reparto aleatorio de las observaciones. La diferencia entre entre métodos suele ser la forma en la que se generan los subconjuntos de entrenamiento/validación (Rodrigo, 2011).

En particular, cuando se tienen tamaños de datos no muy grandes se recomienda emplear ***k_Fold-Cross-Validation*** como método de validación de modelos. Está en capacidad de disminuir la varianza en la estimación del error esperado sin que el coste computacional sea muy elevado respecto a otros métodos. Si bien no viene incluido este método con *spaCy 3.0*, esta validación se puede realizar en Python mediante la herramienta "*scikit-learn*"(Pedregosa et al., 2011) usando el método "*sklearn.model_selection.KFold*" que permite hacer los K grupos para la validación. Este método también se puede utilizar para estimación de hiper-parámetros.

4. Método

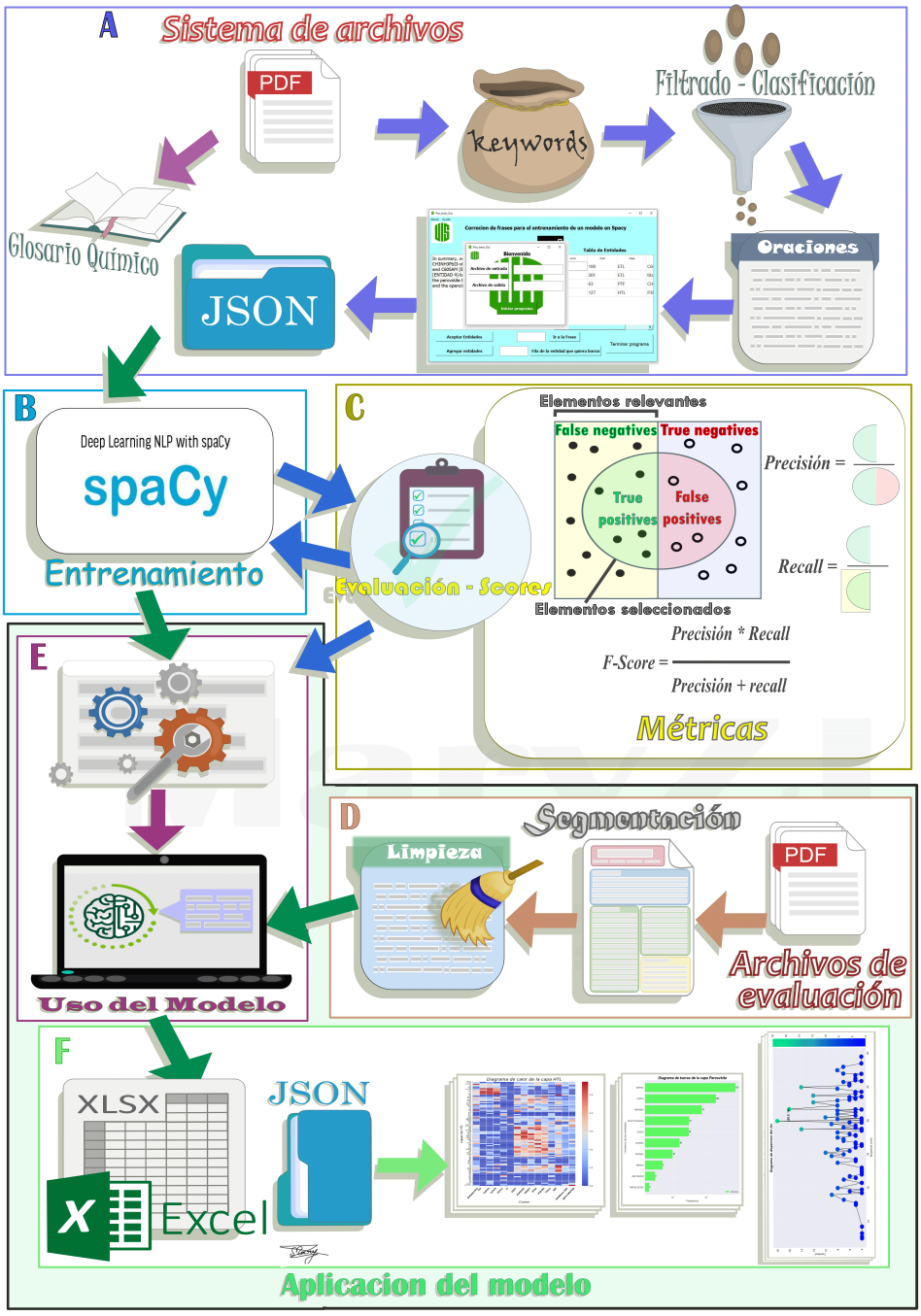

Figura 14. Diagrama esquemático del método usado.

El proceso usado para realizar la extracción automática de parámetros de desempeño de celdas solares de Perovskita, usando la librería de código abierto de lenguaje Python *spaCy 3.0* (Honnibal and Montani, 2015) puede ser resumido en el esquema de la figura 14. Esta figura se divide principalmente en 6 partes fundamentales del proceso, el "**A**"Pre-procesamiento de datos, el "**B**" Entrenamiento del modelo, la "**C**" Evaluación, la "**D**" Extracción de texto de los PDFs, la "**E**" Uso del modelo y la "**F**" visualización de resultados.

4.1. Preprocesamiento de datos

Con el fin de agilizar el proceso a realizar por el modelo *NER* se limito la selección de las entidades nombradas reconocidas y se les asigno una etiqueta o nombre para identificarlas, optando por nombrarlas:

- Entidades nombradas asociadas a las capas de la perovskita: ETL, HTL y PTF (Perovskita).
- Entidades nombradas asociadas a los parámetros de desempeño: VOC, PCE, JSC y FF.
- Entidades nombradas asociadas a los valores tomados por los parámetros de desempeño: NVOC, NPCE, NJSC y NFF.

4.1.1. Limpieza del texto y generación del vocabulario. Los PDFs contienen diferentes codificaciones ASCII extendida (IBM, 2014) que pueden depender de la revistas científicas en las que se publiquen, llevando a errores a la hora de extraer tanto de manera manual como automática del texto y al guárdala en formato tanto texto plano (.txt) como .JSON.

Los archivos científicos con temática de celdas solares de Perovskita contienen tanto nomenclatura especifica de la materia, como la propensión de cambiar y crear nombres específicos

solo usados en el archivo donde se generaron. Para resolver estos problemas se crea un archivo en formato .JSON de símbolos y palabras que necesitan ser cambiadas para estandarizar la nomenclatura química, tomar solo nombres generales de cada compuesto y mejorar el análisis de los archivos, como el que se ve en la Figura 15.


```
{
"FAPbI 3":"FAPbI 3","CH3NH3PbI(3□x)Clx": "CH3NH3PbI3-xClx",
"SpiroOMeTAD"    :"spiro-OMeTAD","~":"","\u0301":""
}
```


Figura 15. Ejemplo de parte del archivo de reemplazos que permite realizar los remplazos necesarios para limpiar el texto.

4.1.2. Creación de conjunto de datos. Usando un "*sistema de archivos*" o base de datos de PDFs, se decide introducir las oraciones y entidades nombradas necesarias para entrenar al modelo *NER*, donde las oraciones se separaron aproximadamente en el 85 % con entidades nombradas y 15 % sin entidades nombradas. Se decide usar la base de datos dividida en aproximadamente en el 90 % para el entrenamiento, las pruebas y la validación, y un 10 % para la evaluación del modelo.

4.1.2.1. Extracción manual y primer versión de los archivos de entrenamiento y evaluación. Las oraciones se seleccionan de forma manual, se limpian y se guardan en una lista de texto plano. Luego con el modelo pre-entrenado liviano de "*spaCy 3.0*" para el idioma ingles usando los dos métodos descritos en la sub-sección 3.3.2, se extraen las entidades nombradas seleccionadas de las oraciones y se organizan en un archivo Json, por ejemplo la oración de la Figura 16.

```
["Before the passivation, the cell had a Jsc of 18.4 mA/cm2,
  Voc of 0.90 V, FF of 0.68, and PCE of 11.3 %.",
    {"entities": [[59,62,"VOC"],[39,42,"JSC"],[90,93,"PCE"],
                  [74,76,"FF"],[66,72,"NVOC"],[46,53,"NJSC"],
                  [97,103,"NPCE"],[80,85,"NFF"]]
    }
],
```

Figura 16. Ejemplo de una oración con sus entidades nombradas en el archivo .JSON para entrenamiento.

4.1.2.2. Depuración de entidades nombradas. El proceso semiautomático realizado en la sub-sección anterior es poco confiable al permitir la aparición de entidades nombradas repetidas, el mal etiquetado de algunas entidades nombradas, deficiencias en el posicionamiento de entidades nombradas, errores en tokenizador, entre otras. Demostrando la necesidad de entrenar un nuevo modelo para aplicaciones especificas. Entonces se debe realizar una depuración del conjunto de datos de entrenamiento y evaluación, siguiendo nuevamente las recomendaciones dadas por la librería "*spaCy 3.0*" en la sub-sección 3.3.2.

4.2. Entrenamiento del modelo

Obtener un modelo óptimo requiere primero encontrar los hiper-parámetros y seguir los procedimientos de regularización de la sub-sección 3.3.4 y de re-muestreo (resampling) de la sub-sección 3.3.5, mediante los métodos que brinda "*spaCy 3.0*" y la librería "*scikit-learn*" (Pedregosa et al., 2011). Solo después de obtener resultados satisfactorios se puede proceder con el entrenamiento y generación final del modelo.

El ajuste de las iteraciones o ***parada temprana*** se puede realizar aplicando el ciclo de entrenamiento a un modelo y observando durante las iteraciones la función de perdidas "*losses*" hasta

que estabilice su valor o cuando el error de la fase de pruebas aumente en vez de disminuir, tomando ese numero "N" de iteraciones como el que debe ser usado. En la ***regularización*** se realizan diferentes entrenamientos variando la ***tasa de olvido (Dropout)*** contenida en la función "*drop*", que tiene como valor por defecto 0.1 y se observan los resultados de las métricas de evaluación (sub-sección 4.3) de cada modelo para decidir el valor final. Por ultimo la ***k-Fold-Cross-Validation*** consiste en usar el método "*sklearn.model_selection.KFold*" para dividir los datos de forma aleatoria en k grupos de aproximadamente el mismo tamaño, k-1 grupos se emplean para entrenar el modelo y uno de los grupos se emplea como validación. Este proceso se repite k veces utilizando un grupo distinto como validación en cada iteración. El proceso genera k estimaciones del error cuyo promedio se emplea como estimación final (Rodrigo, 2011).

4.2.1. Ciclo de entrenamiento. Como se describió en la Figura 12 de la sub-sección 3.3.3 el ciclo de entrenamiento de un modelo "*NER*" de *spaCy 3.0* requiere de la creación de un ciclo de predicción y comprobación de las oraciones con sus entidades nombradas y el uso de un modelo base. En el Pseudocódigo 1, se usa un ciclo de "N" iteraciones del método de *Parada temprana*, el valor "D" de "*drop*" del método de *tasa de olvido (Dropout)* y un numero "B" del tamaño de los lotes para entrenar en cada iteración. Por ultimo se adicionaron todos los demás pipelines que se consideren relevantes del el modelo base, para ayudar en el desarrollo de la sección 4.4.1 .

Pseudocódigo 1 Entrenamiento del modelo.

Require: Numero de ciclos, numero de lotes, la tasa de olvido y el archivo .JSON de entrenamiento.
Ensure: Modelo entrenado con la librería spaCy 3.0.

```
1:  function TRAIN(frases, ciclos)
2:      modelo ← nuevo modelo en blanco
3:      spacy ← modelo base proporcionado por spaCy 3.0
4:      modelo.ner ← cargar el pipeline NER para el modelo en blanco desde spaCy3.0
5:      modelo.ner.entidades ← inicializar todas las nuevas entidades nombradas
6:      for 0:ciclos do
7:          Reorganizar de forma aleatoria el orden de los elementos del archivo frases.
8:          Crea grupos de elementos de tamaño lotes de los elementos reorganizados.
9:          for cada lote en lotes do
10:             for Cada frase y sus entidades nombradas en el grupo lote do
11:                 F ← Frase tomada
12:                 E ← lista de entidades nombradas tomada
13:                 Predice las entidades nombradas de la frase y las compara con las ingresadas.
14:                 P ← lista de entidades nombradas predichas
15:                 Aplica la tasa de olvido (drop).
16:                 Calcula las perdidas del ciclo de entrenamiento (losses).
17:                 Actualiza el modelo.
18:             end for
19:         end for
20:     end for
21:     modelo.pipelines ← cargar los demás pipeline para el modelo
22:     modelo ← Guarda los cambios en el modelo modelo
```

4.2.2. Ajustes del modelo: pruebas, correcciones y validación. Teniendo el ciclo de entrenamiento del modelo definido, se debe obtener los valores “N”,“B” y “D”. Para esto se realiza una serie de entrenamientos de modelos, evaluaciones (sub-sección 4.3) y validaciones en ciclo hasta obtener los mejores resultados posibles de las métricas de evaluación (sub-sección 4.3) y de la validación cruzada. Entonces el procedimiento completo antes de poder realizar el entrenamiento final puede ser descrito en siete fases, Figura 17:

1. Tomamos unos valores iniciales para los parámetros “N” de iteraciones y “B” de lotes del

método *Parada temprana*, y “D” de drop del método *tasa de olvido (Dropout)*, descritos al inicio de esta sección (sección 4.2).

2. Se realiza el entrenamiento del modelo “X”, con los datos de entrenamiento obtenidos mediante la sub-sección 4.1.2..

3. Se evalúa el modelo “X” como se describe en la sub-sección 4.3 con los datos de evaluación, respectivamente, obtenidos mediante la sub-sección 4.1.2. Según se crea tener suficientes modelos con valores aceptables y/o satisfactorios de evaluación, de tiempo de entrenamiento y de la función de perdidas “*losses*” durante el entrenamiento, se pasa al paso 4 o al 5.

4. Según los valores obtenidos de la evaluación, los resultados del tiempo de entrenamiento y de la función de perdidas “*losses*” durante el entrenamiento, se varían los parámetros “N” de iteraciones, “B” de lotes y “D” de drop y volvemos paso 2.

5. Se escoge el modelo con mejor desempeño o desempeño idóneo entre el grupo entrenado, teniendo en cuenta el tiempo de entrenamiento, el valor final de la función de perdidas “*losses*”, y el valor de la media y la varianza de las métricas de evaluación (sub-sección 4.3).

6. Escogido el modelo, se le aplica la Validación cruzada por *K_fold* (k_fold cross validation) descrita al inicio de esta sección (sección 4.2). Dependiendo de los resultados de la validación , se decide si es posible pasar al paso 7 o si es necesario empezar nuevamente el proceso en el paso 4.

7. Como ultimo paso, se entrena un nuevo modelo con los parámetros(“N”,“B” y“D”) del mo-

delo antes escogido (paso 4) y se evalúa. Este modelo y su evaluación, son los resultados finales.

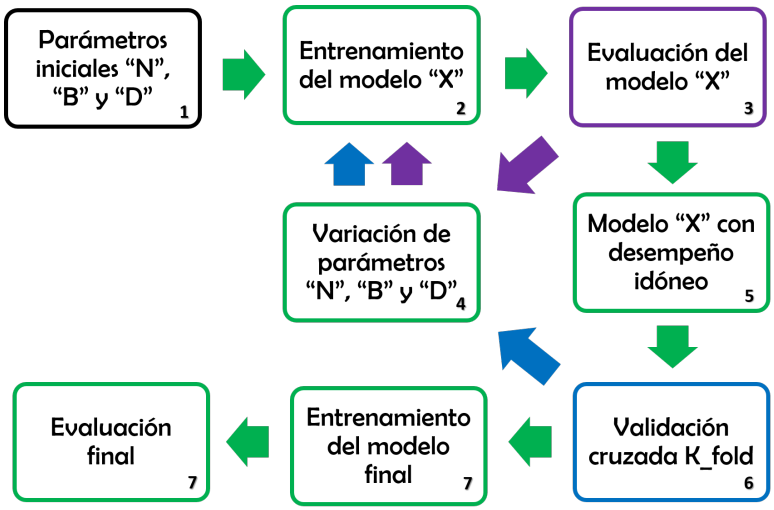

Figura 17. Diagrama de bloques para el procedimiento cíclico de entrenamiento, corrección, validación y evaluación, para obtener los parámetros óptimos “N”,“B” y“D”, para poder entrenar y evaluar el modelo final.

4.3. Evaluación del modelo

En la evaluación de un modelo, primero se toman oraciones o conjuntos de datos con sus entidades nombradas verdaderas, obtenidas por procedimiento de la sub-sección 4.1.2 pertenecientes a los datos de entrenamiento y/o datos de evaluación y se les aplica uno de los métodos de evaluación de *spaCy 3.0* descritos en la sub-sección 3.3.5.

La evaluación del modelo descrita en la sub-sección 3.3.5 con el método “*scorer*” sigue una estructura como la expuesta en el Pseudocódigo 2, que funciona ingresando una o unas oraciones y/o texto junto a sus entidades nombradas verdaderas (elementos relevantes) y un modelo. Al modelo ingresado se le aplica las oraciones y/o texto y se obtiene sus supuestas entidades nombradas

(elementos seleccionados) y luego se realiza la evaluación de las métricas generando un archivo Excel(.xlsx) donde se guardan los valores encontrados.

Pseudocódigo 2 Evaluar el modelo usando spaCy 3.0.

Require: Archivo con oraciones y/o texto con sus entidades nombradas reconocidas.
Ensure: Archivo excel con la evaluación del modelo.

```
1: function SCORE(Modelo, Frases, Archivo)
2:     oraciones ← Oraciones y/o texto de Frases
3:     EnV ← Entidades nombradas verdaderas de las oraciones y/o texto de Frases
4:     for cada oracion en oraciones do
5:         Ent ← Aplica el modelo Modelo a oracion y extrae las entidades nombradas
6:         Precision, recall, fcore ← Aplica las 3 métricas a oracion usando spaCy 3.0, EnV y Ent
7:     end for
8:     Excel ← Guarda Precision, recall, fcore en un excel con el nombre Archivo
```

4.4. Aplicación del modelo

4.4.1. Extracción de texto de los PDFs. Para la extracción de información de parámetros de desempeño de los PDFs tal como se plantea en este trabajo es necesario de extraer automáticamente primero el texto de interés de los PDFs seleccionados y codificarlo (usando lo descrito en la sub-sección 4.1.1), haciendo posible la aplicación del modelo entrenado .

En la extracción de los textos se usa la librería de python *PyMuPDF* y se observa la distribución promedio de la información en los artículos científicos de celdas solares de Perovskita, reduciendo la extracción de texto cuando es posible a solo secciones especificas con la mayor probabilidad de contener los parámetros de interés. Las secciones de los PDFs seleccionadas son las correspondientes a los “abstract”, los “resultados”, la “discusión”, las “conclusiones” o semejantes.

Para permitir extraer solo las secciones seleccionadas se hace uso de expresiones regulares y de la información de estilos, fuente y tamaño suministrados por *PyMuPDF* de los PDFs como en

el ejemplo la Figura 18. En caso de lograr extraer una o ninguna sección, se pasa a extraer el texto plano de todo el PDF.

```
{
'number': 0, 'type': 0,
'bbox': (60.49129867553711, 117.61646270751953, 507.4458312988281, 154.93972778320312),
'lines': [
    {'spans': [{'size': 16.999900817871094, 'flags': 4, 'font': 'AdvOTce3d9a73',
        'color': 0,'ascender': 0.8849999904632568, 'descender': -0.25,
        'text': 'High-Performance Perovskite-Polymer Hybrid Solar Cells via',
        'origin': (60.49129867553711, 132.661376953125),
        'bbox': (60.49129867553711, 117.61646270751953, 507.4458312988281,
        136.91134643554688)}], 'wmode': 0, 'dir': (1.0, 0.0),
        'bbox': (60.49129867553711, 117.61646270751953, 507.4458312988281,136.91134643554688)},

    {'spans': [{'size': 16.999900817871094, 'flags': 4, 'font': 'AdvOTce3d9a73', 'color': 0,
        'ascender': 0.8849999904632568, 'descender': -0.25,
        'text': 'Electronic Coupling with Fullerene Monolayers',
        'origin': (60.49129867553711, 150.68975830078125),
        'bbox': (60.49129867553711, 135.6448516845703, 407.9981994628906, 154.93972778320312)}],
        'wmode': 0, 'dir': (1.0, 0.0),
        'bbox': (60.49129867553711, 135.6448516845703, 407.9981994628906, 154.93972778320312)}
    ]
}
```

Figura 18. Ejemplo de la matriz de con la información y el texto que se obtiene con PyMuPDF, para el titulo de un PDF.

4.4.2. Uso del modelo. Conociendo ya la estructura normal de las oraciones que contienen los valores de los parámetros de desempeño de las celdas solares de Perovskita, y teniendo el modelo NER entrenado para reconocer los tres tipos de entidades nombradas previamente descritos al inicio de este capitulo (sección 4.1), aun no podemos tener certeza de la asociación entre las entidades nombradas cuando el texto esta conformado de mas de una oración, como es el caso de las secciones de un PDF.

Para minimizar los problemas y mejorar la extracción de información y así poder aplicar mejor el modelo a nuevos datos, se usa una estructura como la del Pseudocódigo 3, que guarda de forma ordenada las entidades nombradas con su oración de origen, y relaciona las entidades nombradas asociadas a los parámetros (PCE, VOC, JSC y FF) con las entidades nombradas asociadas

a los valores tomados por los parámetros (NPCE, NVOC, NJSC y NFF).

Pseudocódigo 3 Corrección y optimización de la extracción.

Require: Modelo entrenado y el PDF a extraer información.
Ensure: Diccionario con la información de las capas, los parámetros y las oraciones extraídas.

```
1:  function TEXT_EXTRACTION(Modelo, PDF)
2:      secciones ← SEGMENTATION(PDF) ▷ Llama el procedimiento de sub-sección 4.4.1.
3:      for cada seccion en secciones do
4:          Texto ← CLEAN(seccion)          ▷ Limpia, codifica y estandariza, sub-seccion 4.1.1.
5:          for Cada oracion en Texto do
6:              Entidad1 ← Entidades nombradas asociadas a los parámetros, con Modelo
7:              Entidad2 ← Entidades nombradas asociadas a los valores de los parámetros, con Modelo
8:              Entidad3 ← Entidades nombradas asociadas a las capas, con Modelo
9:              Diccionario ← Guarda entidad3 en el diccionario
10:             if entidad1 existe en oracion then
11:                 if entidad2 es un numero y es mayor a 0.0 then
12:                     Diccionario ← Guarda entidad2 en el diccionario
13:                 end if
14:             end if
15:             Diccionario ← Guarda oracion en el diccionario
16:         end for
17:     end for
```

4.4.3. Obtención y visualización de resultados. Los resultados del uso del modelo para una carpeta o PDF que se desean analizar, son guardados en una carpeta. La carpeta esta conformada por archivos de Excel (.xlsx), un archivo de meta-datos en formato .JSON y dos archivos de información sobre el proceso en formato de texto plano (.txt). Adicionalmente también se guarda la información en los archivos Excel (.xlsx) de las tablas que se logran extraer mediante los códigos descritos en la sección 7.2.

Guardados todos los resultados se procede a realizar nuevamente una limpieza y estandarización de los datos extraídos para su visualización mediante mapas de calor, diagramas de barras y diagramas de dispersión, que serán guardados en formato .png. La limpieza se hará usando el pro-

cedimiento de la sub-sección 4.1.1 modificado para caso del archivo .JSON de resultados, ejemplo en la Figura 19. Para la estandarización se usara el modulo "*difflib*" de python y el conocimiento que se tiene gracias a los artículos científicos de celdas solares de Perovskita, de las entidades nombradas asociadas a las capas (HTL, ETL y Perovskita) mas comunes, las asociadas a los parámetros de desempeño y las asociadas a los valores de los parámetros de desempeño (PCE, JSC, VOC, FF, NPCE, NJSC, NVOC y NFF). Por ultimo para gráficar los mapas y diagramas de visualización se usara el modulo *matplotlib.pyplot* de python.


```
{
"C60-SAM":"C60SAM","TiO2-nanorod":"TiO2","mTiO2":"TiO2",
"mZnO":"ZnO","Ni-Fe2O3":"Fe2O3","CH3NH3PbI3-xClx":"MAPbI3-xClx",
"Spiro-O-MeTAD":"spiro-OMeTAD","znO":"ZnO"
}
```


Figura 19. Ejemplo de parte del archivo de reemplazos que permite realizar los cambios necesarios para la limpieza de los resultados.

5. NLP aplicado a Perovskitas

Se aplica el proceso descrito en el método, Figura 20, tomando la base de datos de 582 PDFs descargados de diferentes artículos científicos sobre celdas solares de perovskita convencionales, publicados entre 2013 y 2018 consignados en el conjunto de datos de Çağla Odabaşı and Yıldırım (2019). Y se genera la herramienta de software para la extracción de parámetros de desempeño en celdas solares de Perovskita.

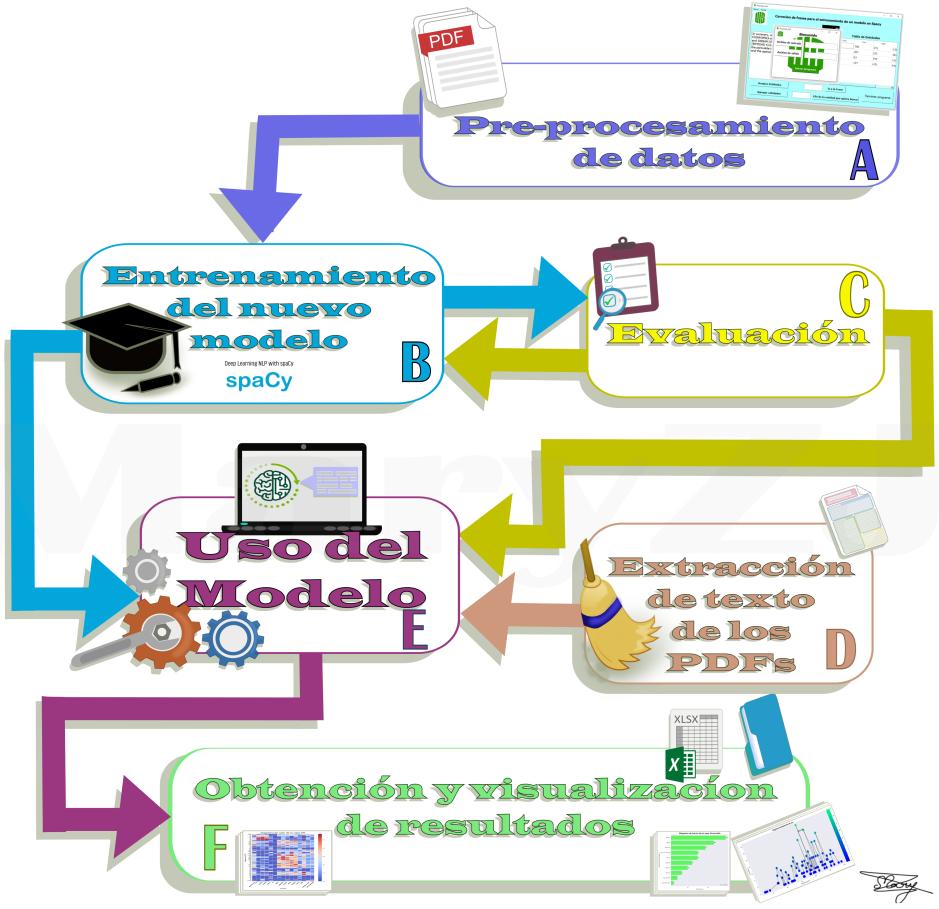

Figura 20. Diagrama de bloques que resume la metodología expuesta en la sección 4.

Dentro de la carpeta "*NerPSC*"se encuentran los scripts "*Module.py*" y "*Process.py*" para compilación en linea de comandos e importar las funciones del proyecto, respectivamente. Se crea la sub-carpeta "*Modulo_NerPSC*" para contener todos los scripts (códigos) de funcionamiento, se guarda la base de datos en la sub-carpeta "*data_pdf*", se guardan todos los resultados tanto fuera como en la sub-carpeta "*DATA*", y se guardan los modelos de validación y el modelo final "*Model_NerPSC*". En la sub-carpeta "*Modulo_NerPSC*" se encuentra en particular el script "*Functions.py*" el cual contiene funciones que se utilizan en diferentes partes del proceso.

5.1. Generación del modelo NER

5.1.1. Creación del conjunto de datos. Para empezar con la creación del conjunto de datos como se describió en la sección 4.1.1, primero se usan unas de las funciones guardadas en el script "*Functions.py*" que permiten aplicar un proceso de limpieza del texto plano extraído. Con esto se obtienen datos mas limpios y entendibles como se observa en el ejemplo de la Figura 21, en la cual se muestra la comparación entre la apariencia de dos oraciones, cuando se extraen en texto plano, se convierten a formato .JSON y luego cuando se les a aplicado la limpieza.

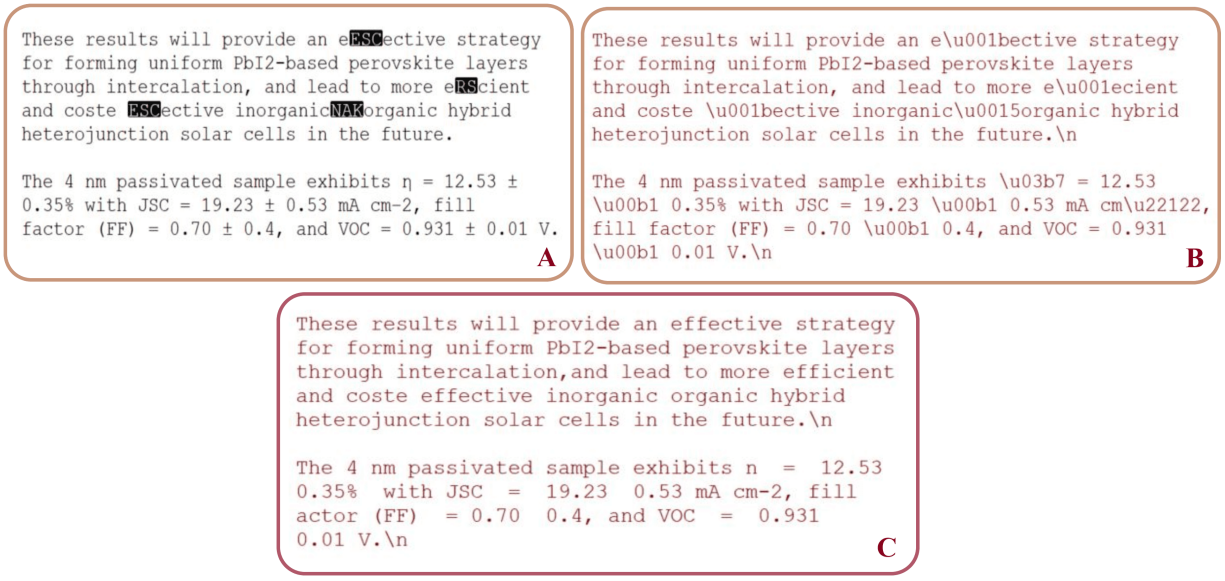

Figura 21. Evolución de dos frases ejemplo en el proceso de limpieza del método de la sección 4.1.1. En donde A son las oraciones en texto plano(.txt), B las oraciones al pasar a formato .JSON sin limpieza y C las oraciones en formato .JSON luego del proceso de limpieza.

Se escogen 2182 oraciones manualmente para entrenamiento y evaluación dentro de las cuales 269 no contienen entidades nombradas, se les aplica la limpieza, se guardan en los archivos "*Train.txt*" y "*Evaluation_phrases.txt*", respectivamente y se realiza los procedimientos expresados en las secciones 4.1.2.1 y 4.1.2.2.

Primero para el procedimiento de la sección 4.1.2.1 se usa el script "*Sentences.py*" junto a funciones del script "*Functions.py*", obteniendo el archivo "*Train.json*" con 2059 oraciones y 6514 entidades nombradas para el entrenamiento y el archivo "*Evaluation_phrases.json*" con 123 oraciones y 726 entidades nombradas para la evaluación.

Luego para la sección 4.1.2.2 se usa la "*API*" llamada "*Gui.py*" guardada en la sub-sub-carpeta "*./NerPSC/Modulo_NerPSC/GUI*" (Figuras 28, 29 y 30), en fase alfa codificada en

Python descrita en la sección 7.1 , como alternativa a la "herramienta de anotación moderna para crear datos de entrenamiento para modelos de aprendizaje automático" de paga "*prodigy*" (Honnibal and Montani, 2021b) del mismo desarrollador de "*spaCy 3.0*", que nos permite ingresar los datos e ir gráficamente modificando, limpiando las entidades nombradas del archivo .JSON y luego guardándolas en un nuevo archivo. Obteniendo el archivo final del grupo de entrenamiento "*TrainPOS.json*" y del grupo de evaluación
"*Evaluation_phrasesPOS.json*".

5.1.2. Entrenamiento, ajustes y validación del modelo. Para el procedimiento de la sección 4.2 se usan los scripts "*Training.py*" y "*Score.py*", unas funciones del script "*Functions.py*" y el archivo "*TrainPOS.json*" obtenido en el procedimiento de la sub-sección anterior.

Primero siguiendo el ciclo de retroalimentación expresado en la sub-sección 4.2.2, se asumen como parámetros iniciales de $N = 100$ para los Ciclos, $B = 1$ para lotes y $D = 0.5$ para el drop, y se entrenan 19 modelos diferentes (Anexo 3) usando la función "*TRAIN*" del script y se evaluaron con la función "*SCORE*" y el archivo "*Evaluation_phrasesPOS.json*".

Teniendo los 19 modelos se procede a escoger el que fue considerado como el mejor, para esto se realiza la Tabla 7(Anexo 3) y las gráficas de las Figuras 33, 34 y 35 (Anexo 3), donde se observan los resultados de las métricas de evaluación para cada modelo y su tiempo de entrenamiento (T). Luego se elige tomar y guardar el modelo 16 (*m16*) con parámetros de entrenamiento $N = 85$, $B = 100$ y $D = 0.2$, su archivo de entrenamiento "*m16_Training_output.txt*" y sus archivos de evaluación "*SCOREm16e.xlsx*" y "*m16_Score_output.txt*".

Luego al modelo "*m16*" le aplicamos la validación cruzada K_fold para comprobar la elec-

ción, usando la función "*KFOLD_validation*" del script. Se realiza la validación con $K = 5$ que tarda 4 horas 17 minutos y 9 segundos, obteniendo valores de media(mean) para la Precisión, Recall y el F-score de 94.05 %, 95 % y 94 % con una varianza de 2.5 %, 2.1 % y 2.1 %, respectivamente. Se guardan los modelos K_fold como "*m16_x*"($x = [0 : K-1]$) y los archivos "$m16\_Kfold - Validation.xlsx$" con los resultados de la validación y "$m16\_Kfold - validation\_output.txt$" con información del proceso de la validación.

Por ultimo con los resultados satisfactorios de validación cruzada y los parámetros del modelo "*m16*"($N = 85, B = 100$ y $D = 0.2$) se entrena el modelo NER final llamado "*Model_NerPSC*", que tarda 1 hora 4 minutos y 17.2 segundos. Se guarda el archivo del modelo y su archivo de proceso de entrenamiento "$Model\_NerPSC\_Training\_output.txt$".

También terminado el proceso se aplicaron unas frases de prueba al modelo inicial(entrenado con los valores iniciales) y al modelo final "*Model_NerPSC*" (entrenado con los valores mas óptimos encontrados), para observar las diferencia visuales, como se ve el ejemplo de la Figura 22.

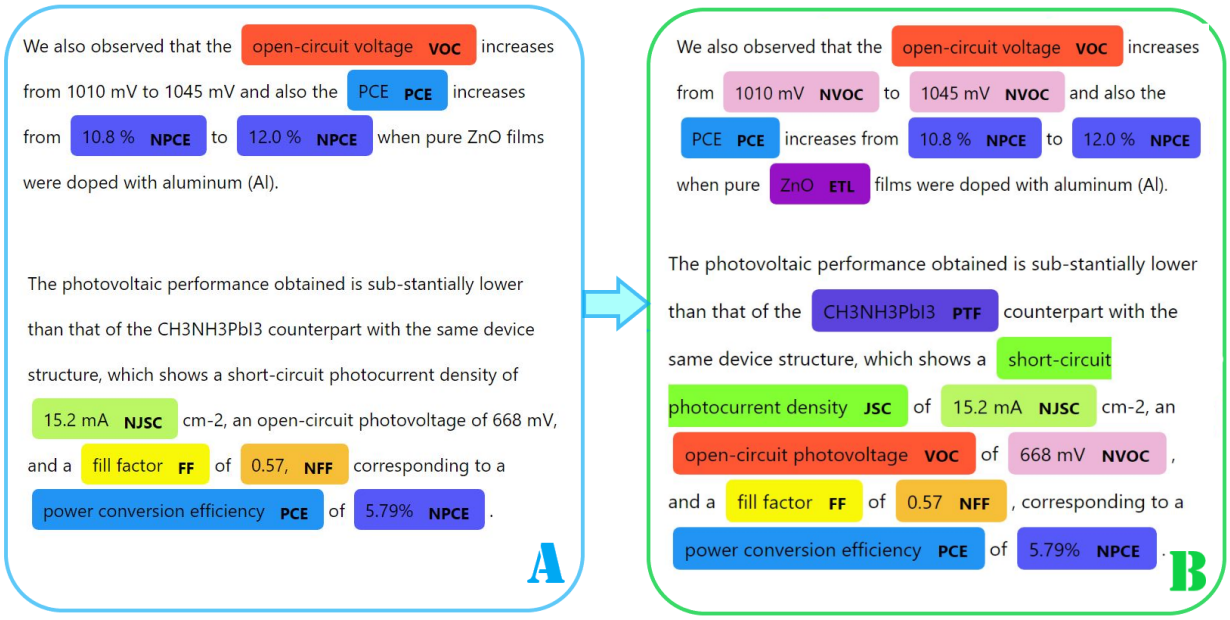

Figura 22. Ejemplo de las mejoras obtenidas con las pruebas y correcciones en la extracción de entidades nombradas del modelo NER. Donde A es el modelo inicial salido del procedimiento de la sección 5.1.2 y B es el modelo luego del procedimiento de la sección 5.2.

5.1.3. La evaluación del modelo. Por ultimo se realiza el procedimiento de evaluación al modelo "*Model_NerPSC*" como se describe en la sección 4.3, usando el script "*Score.py*" y el archivo "*Evaluation_phrasesPOS.json*", que resulta en valores de media para la Precisión, Recall y el F-score de 93.3 %, 94.5 % y 93.4 % con un varianza de 1.4 %, 1.4 % y 1.2 %, respectivamente (ver Tabla 3 y 4). Los resultados se guardan en los archivos "*SCOREModel_NerPSCe.xlsx*" y "*Model_NerPSC_Score_output.txt*"

Tabla 3

Tabla resumen de valores obtenidos al efectuar la evaluación del modelo Model_NerPSC.

Metrics	Presicion	Recall	F-score
count	123	123	123
mean	0,933655	0,945483	0,934181
std	0,118921	0,11992	0,1105
min	0,333333	0,333333	0,333333
25 %	0,888889	0,909091	0,888889
50 %	1	1	1
75 %	1	1	1
max	1	1	1
var	0,014027	0,014264	0,012111

Tabla 4

Tabla de valores obtenidos al efectuar la evaluación del modelo Model_NerPSC por entidad.

Metrics	ETL_p	ETL_r	ETL_f	HTL_p	HTL_r	HTL_f
count	37	37	37	28	28	28
mean	0,918919	0,887387	0,896396	0,755952	0,785714	0,766667
std	0,276725	0,299315	0,286936	0,416931	0,417855	0,413954
min	0	0	0	0	0	0
25 %	1	1	1	0,625	1	0,766667
50 %	1	1	1	1	1	1
75 %	1	1	1	1	1	1
max	1	1	1	1	1	1

Metrics	PTF_p	PTF_r	PTF_f	PCE_p	PCE_r	PCE_f
count	31	31	31	92	92	92
mean	0,956989	0,967742	0,96129	0,847826	0,869565	0,855072
std	0,187418	0,179605	0,181984	0,345609	0,338627	0,339878
min	0	0	0	0	0	0
25 %	1	1	1	1	1	1
50 %	1	1	1	1	1	1
75 %	1	1	1	1	1	1
max	1	1	1	1	1	1

Metrics	VOC_p	VOC_r	VOC_f	JSC_p	JSC_r	JSC_f	FF_p	FF_r	FF_f
count	43	43	43	52	52	52	49	49	49
mean	0,976744	0,976744	0,976744	0,878205	0,875	0,871154	0,979592	1	0,986395
std	0,152499	0,152499	0,152499	0,308053	0,311149	0,305507	0,099957	0	0,066638
min	0	0	0	0	0	0	0,5	1	0,666667
25 %	1	1	1	1	1	1	1	1	1
50 %	1	1	1	1	1	1	1	1	1
75 %	1	1	1	1	1	1	1	1	1
max	1	1	1	1	1	1	1	1	

5.2. Aplicación del modelo

5.2.1. La extracción del texto de los PDFs. Para realizar el proceso de la sección 4.4.1 se usa el script "*Segmentation.py*" y funciones del script "*Functions.py*", que permiten la extracción, limpieza y segmentación del texto del PDF para obtener las secciones seleccionadas obteniendo un diccionario con el nombre de la sección, el texto contenido y su información de posición en el PDF, como se ve en el ejemplo de la Figura 23.

```
{   'ABSTRACT':

        ['ABSTRACT: A plethora of solution-processed materials have been developed for solar cell applications.
        Hybrid solar cells based on light absorbing semiconducting polymers infiltrated into mesoporous TiO2
        are an interesting concept, but generating charge at the polymer-metal oxide heterojunction is challenging.
        Metal-organic perovskite absorbers have recently shown remarkable efficiencies but currently lack the range
        of color tunability of organics. Here, we have combined a fullerene self- assembled monolayer (C60SAM)
        functionalized mesoporous titania, a perovskite absorber (CH3NH3PbI3-xClx), and a light absorbing polymer
        hole-conductor, P3HT, to realize a 6.7% efficient hybrid solar cell. We find that photoexcitations in both
        the perovskite and the polymer undergo very efficient electron transfer to the C60SAM. The C60SAM acts as an
        electron acceptor but inhibits further electron transfer into the TiO2 mesostructure due to energy level
        misalignment and poor electronic coupling. Thermalized electrons from the C60SAM are then transported through
        the perovskite phase. This strategy allows a reduction of energy loss, while still employing a "mesoporous
        electron acceptor", representing an exciting and versatile route forward for hybrid photovoltaics incorporating
        light-absorbing polymers. Finally, we show that we can use the C60SAM functionalization of mesoporous TiO2 to
        achieve an 11.7% perovskite-sensitized solar cell using Spiro-OMeTAD as a transparent hole transporter.
        KEYWORDS: Perovskite, fullerene, hybrid photovoltaics, self-assembled monolayer, photoinduced absorption,
        photoluminescence ',
        (69.44879913330078, 255.65863037109375, 537.2911376953125, 445.73455810546875)],

    'In summary,':

        ['The relative increase in performance with the C60SAM is reduced compared to P3HT devices. The open-circuit
        voltage is only slightly increased, suggesting that the addition of lithium additives to the hole transporter,
        which is required to improve spiro-OMeTAD conductivity, enables a certain degree of forward electron transfer to
        the titania.9,22The improved fill factor suggests that there is reduced recombination under working conditions.
        In summary, we have demonstrated that, by using CH3NH3PbI3-xClx perovskite as an absorber and C60SAM as an
        interlayer in a P3HT-based hybrid device, electron transfer from the perovskite to the titania can be blocked
        and the open- circuit voltage loss reduced. The C60SAM acts as a very effective electron acceptor from the
        perovskite and the P3HT polymer, additionally providing polymer photoactivation. Notably, although long-range
        electron transport and photocurrent collection occurs through the perovskite, the electrons are mostly mediated
        through the C60SAM, regardless of whether they originate from light absorbed in the perovskite or P3HT. We were
        able to fabricate devices with power conversion efficiencies exceeding 6.7% and open-circuit voltages over 0.8 V,
        compared to devices with □4% efficiency and □0.7 V without C60SAM functionalization. This strategy can allow
        enhanced electronic coupling between perovskites and polymer semiconductors and hence represents an exciting
        route forward for hybrid photovoltaics. The architecture was also used with Spiro-OMeTAD as the hole transporter
        to produce hybrid solar cells with power conversion efficiencies of 11.7%. Finally, the strategy offers a novel
        and versatile system to study charge transport within a multitrapping framework, as well as other fundamental
        photovoltaic phenomena. ',
        (324.4534912109375, 685.4866943359375, 334.4375, 709.1347045898438)]
}
```

Figura 23. Ejemplo del diccionario obtenido con secciones relevantes de un PDF.

5.2.2. Uso y Optimización. Terminado todo el proceso asociado al modelo en la sección 5.1 y la extracción del texto en la sección 5.2.1, se pasa a la adaptación y optimización de la salida del modelo tal como se describe en la sección 4.4.2 mediante el script "*Text_extraction.py*", que permite obtener un diccionario de python con los datos extraídos del PDF como se ve en el ejemplo 24.


```
"a0001.pdf": {
    "CAPAS": {
        "ETL": ["TiO2","alumina","C60SAM","C60","titania","SAM"],
        "HTL": ["spiro-OMeTAD","PCPDTBT","Z. V.","P3HT","Li-TFSI"],
        "PTF": ["CH3NH3PbI3-xClx.1","CH3NH3PbI3","CH3NH3PbI3-xClx"]
    },
    "PARAMETROS": {
        "VOC": ["0.8 V","0.7 V","0.82-0.84 V"],
        "PCE": ["3.8%","4%","6.7%","11.7%"],
        "JSC": [],
        "FF": ["0.65-0.72"]
    },
    "ORACIONES": ["There is a notable increase in the fill factor (0.65-0.72) with the addition of the C60SAM,
        a marginal increase in short-circuit current (19.4-19.6 mA cm-2), and a slight increase in open-circuit
        voltage (0.82-0.84 V).",
        "We were able to fabricate devices with power conversion efficiencies exceeding 6.7%  and open-circuit
        voltages over 0.8 V, compared to devices with  4%  efficiency and  0.7 V without C60SAM functionalization.",
        "The architecture was also used with spiro-OMeTAD as the hole transporter to produce hybrid solar cells
        with power conversion efficiencies of 11.7% .",
        "The power conversion efficiency increases from 3.8%  to 6.7%  in the presence of the C60SAM
        functionalization, with an increase in both open-circuit voltage (0.68-0.81 V) and short-circuit current
        (10.1-14.9 mA cm-2)."
    ]
}
```


Figura 24. Ejemplo del diccionario de python con los datos relevantes de un PDF, extraídos en este trabajo.

Además se hace uso del script "*Table_extraction.py*" descrito en la sección 7.2, donde de forma resumida se intenta hacer un acercamiento a la extracción de tablas en textos en formato PDF, con el fin de intentar complementar los datos ya extraídos.

5.2.3. Obtención y visualización de resultados. Se crea el script "*Saved.py*", se usan los scripts de la sub-sección 5.2.1 y 5.2.2 y la carpeta de archivos "*data_pdf*" que contiene la base de datos descrita al inicio del capitulo, para extraer y guardar los resultados que genera la herramienta de extracción de parámetros de desempeño en celdas solares de Perovskita a un grupo

de PDFs. Con el script "*Saved.py*" y los archivos de "*data_pdf*", se obtiene y se guarda dentro de la sub-carpeta llamada "*DATA*" la información obtenida en los PDFs, tal como se describió en la sub-sección 4.4.3 .

En la sub-carpeta "*DATA*" se encuentran tres tipos diferentes de archivos, "*DATA_PDF{A}.xlsx*" (*A* =nombre del PDF), "*DATA.json*", "*DATA_Data_Extraction_output.txt*" y "*DATA_NOT_extract.txt*" relacionados a los resultados de la extracción de los datos y las tablas. En el grupo de archivos .xlsx, se encuentra uno para cada PDF analizado, ejemplo las tablas 5 y 6. Los dos archivos del tipo .txt se usan para contar la cantidad de PDFs que no se logran extraer y para saber el tiempo gastado en el proceso, ejemplo la Figura 25. Por ultimo un archivo del tipo .JSON se usa para guardar la meta-data de la extracción de todos los PDFs, ejemplo la Figura 24.

Además como se describe nuevamente en la sub-sección 4.4.3, usando los recursos dados por Python para la creación de gráficas y el análisis de meta-datos se usan los scripts "*Heat_map.py*", "*Layer_Bar_chart.py*" y "*Parameter_bar_chart.py*" guardados en la sub-carpeta "*./NerPSC/Modulo_NerPSC*", donde se realiza la ultima limpieza y estandarización de los datos extraídos y se generan las gráficas que permitan el mejor análisis de los resultados, como se ve en el siguiente capitulo (sección 5.3.2).

Figura 25. Ejemplo de los archivos .txt, donde A es el archivo con la cuenta y registro de los PDFs a los que no se les pudo aplicar el proceso y B es el archivo con el tiempo necesario para realizar el proceso.

Tabla 5
Tabla ejemplo de la información obtenida con el proceso descrito en el capitulo 7.2 para un PDF.

2Device structure [reference] NRA length	**h/ %**	**Voc/V**	**Jsc/mA cm**	**FF**
12Spiro-MeOTAD/D102/ZnO NRAs 600 nm	0,093	0,47	0,73	—
Spiro-MeOTAD/D102/ZnO– 12MgO/NRAs 600 nm	0,156	0,49	1,12	—
12Spiro-MeOTAD/D102/ZnO–ZrO2/NRAs 600 nm	0,283	0,47	2,14	—
12Spiro-MeOTAD/D149/ZnO NRAs 600 nm	0,088	0,47	0,71	—
12Spiro-MeOTAD/D149/ZnO–MgO/NRAs 600 nm	0,278	0,58	1,53	—
Spiro-MeOTAD/D149/ZnO– 12ZrO2/NRAs 600 nm	0,596	0,57	3,02	—
20P3HT/CdS/ZnO NRAs 800 nm	0,24	0,34	1,6	0.43
20P3HT/CdSe/CdS/ZnO NRAs 800 nm	1,5	0,675	4,2	0.52
21P3HT/N3/ZnO NRAs 250 nm	0,13	0,46	0,72	0.38
15P3HT/Z907/ZnO NRAs 110 nm	0,2	0,3	1,73	0.39
22P3HT/Z907/ZnO NRAs 500 nm	0,2	0,23	2	—
11MEH-PPV/Z907/ZnO NRAs 170 nm	0,61	0,29	6,53	0.32
23P3HT/mercurochrome/ZnO NRAs 300 nm	0,13	0,34	0,91	0.43
18P3HT/CdS/ZnO 180 nm	0,11	0,6	0,39	0.48
19MEH-PPV/CdS/ZnO NRAs 200–300 nm	0,65	0,78	2,87	0.29
13CuSCN/N719/ZnO NRAs 11–12 mm	1,7	0,57	8	—
9Spiro-MeOTAD/Z907/ZnO NRAs-TiO2 50 mm	5,65	0,788	12,2	0.51

Tabla 6

Tabla ejemplo de la información obtenida con el proceso descrito en el capítulo 4 para un PDF.

PDF	ETL	HTL	PTF	VOC	PCE	JSC	FF	Oraciones
a0004.pdf	TiO2	MEH-PPV	0.91 0.43 P3HT	0.25-0.59 V	5.2%	4.23 mA	0.58.	However, after reaching a maximum of 12.7 mA cm:2, Jsc decreases slightly.
a0004.pdf	CdSe	D149	CH3NH3PbI3	0.68 V	4.35%	2.0 mA	–	This trend is similar to that described for ZnO NRA/CdS/CdSe solar cell devices, the difference is that the Jsc in the perovskite solar cell is roughly 3 times higher than that in the CdS/CdSe system (Jsc = 4.23 mA cm:2).20 Jsc is strongly dependent on the nanorod length; it increases from 8.9 to 12.7 mA cm:2 as the nanorod length increases from 400 to 1000 nm.
a0004.pdf	CdS	ZrO2	CH3NH3PbI2Cl	0.75 V	5.0%	8.9	–	The optimized solar cell exhibited an efficiency of 5.0% under 1000 W m:2 AM1.5 illumination.
a0004.pdf	MgO	Z907	0.093 0.47 0.73 - spiro-MeOTAD	0.42 V	–	12.7 mA	–	The Voc decreased from 0.75 V to 0.42 V as the nanorod length increased from 400 nm to 1400 nm.
a0004.pdf	ZnO	spiro-MeOTAD/	–	–	–	–	–	At 10% of this light intensity an efficiency of 5.2% was recorded.
a0004.pdf	Al2O3	2.14 - spiro-MeOTAD	–	–	–	–	–	After 500 hours of storage the overall conversion efficiency was only slightly decreased from a maximum value of 5.0% to 4.35% .
a0004.pdf	–	spiro-MeO-TAD	–	–	–	–	–	Note that the Voc and Jsc values found here are much higher than typical values for dye-sensitized ZnO nanorod array solar cells with a rod length ranging between 200 nm and 1800 nm (Voc = 0.25-0.59 V, Jsc = 0.31-2.0 mA cm:2).14,22
a0004.pdf	–	P3HT	–	–	–	–	–	The top efficiency was 5.0% under 1000 W m:2 AM 1.5 G illumination with a Jsc of 12.7 mA cm:2, a Voc of 0.68 V and a fill factor of 0.58.
a0004.pdf	–	spiro-MeOTAD	–	–	–	–	–	–
a0004.pdf	–	CuSCN	–	–	–	–	–	–
a0004.pdf	–	D102	–	–	–	–	–	–

5.3. Análisis de resultados

5.3.1. El modelo. Todo el procedimiento efectuado en la sección 5.1 da resultados para la Precisión, Recall y el F-score de 93.3 %, 94.5 % y 93.4 % con varianza de 1.4 %, 1.4 % y 1.2 %, respectivamente (ver Tabla 3 y 4) como se ve en la sub-sección 5.1.3. Estos resultados de la evaluación del modelo permiten comprobar que el método usado el fue correcto, puesto que durante los ciclos de entrenamiento-evaluación se encuentra un modelo un alto valor en las métricas de evaluación y con una baja varianza a nuevos datos, cosa que luego se comprueba con la validación cruzada, mostrando que mediante el procedimiento de generación del modelo seleccionado se logro evitar los desajustes.

5.3.2. La extracción de parámetros de despeño sobre celdas solares de Perovskita. Usando los scripts "*Heat_map.py*", "*Layer_Bar_chart.py*" y "*Parameter_bar_chart.py*" como se describió en la sub-sección 5.2.3 se obtienen tres tipos diferentes de gráficas en base a los datos guardados en la carpeta "*DATA*", diagramas de calor y de barras para analizar las capas de la celda solar de Perovskita, y diagramas de dispersión para los parámetros de desempeño de las celdas solares de Perovskita.

Los diagramas de calor se usan para escoger de forma manual la clusterización que permite observar la distribución de los compuestos usados en cada capa, que se pueden observar en el Anexo 4.

Realizada la clusterización, se grafican los diagramas de barras para notar la frecuencia de cada compuesto, como se ven en la Figura 26 o en el Anexo 5. En los Diagramas de barras se

observa que la tendencia para las celdas solares de Perovskita estudiadas en artículos científicos de entre los años 2013 y 2018, esta en una estructura de tipo spiro-OMeTAD/MAPbl3/TiO2.

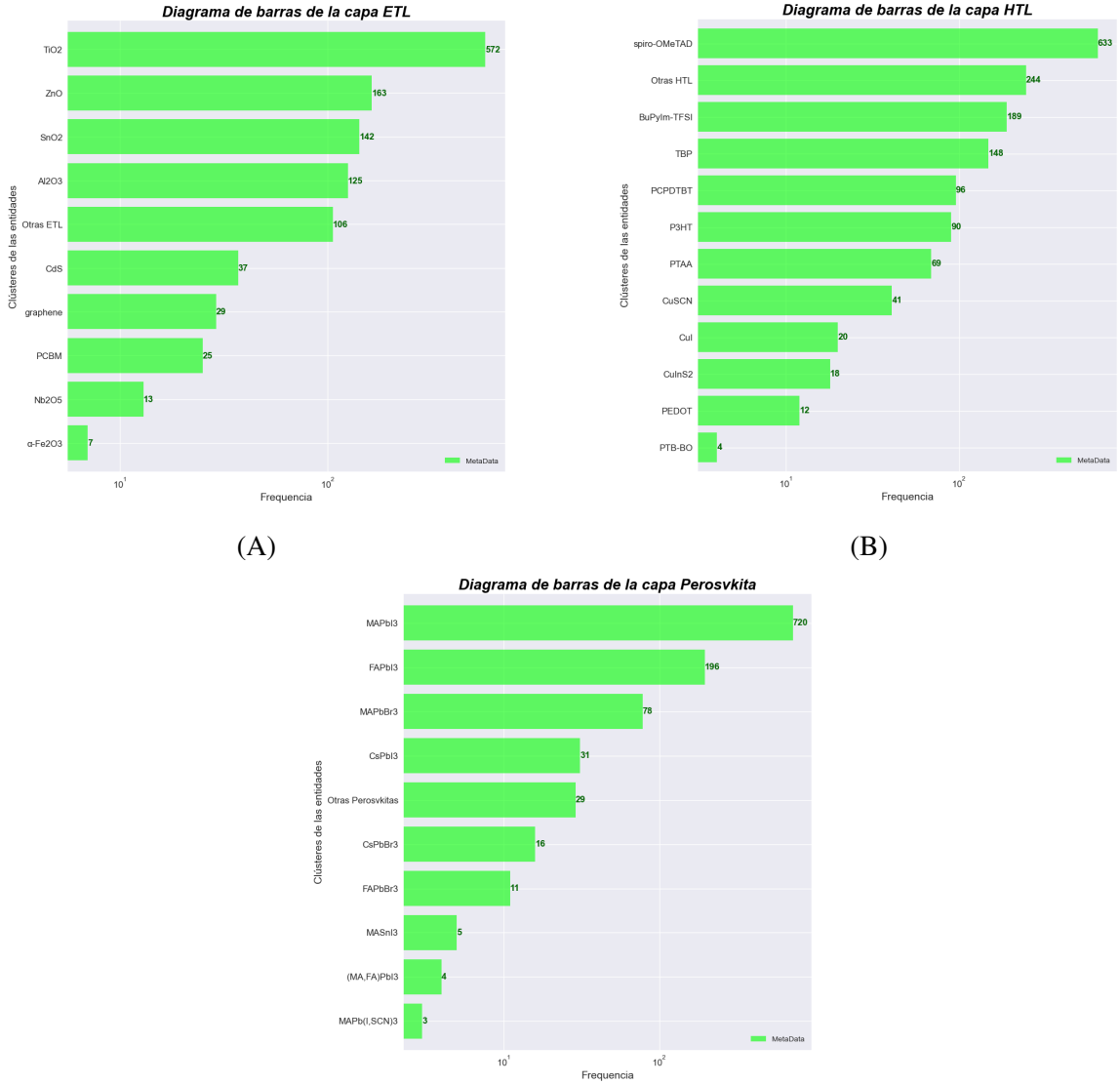

Figura 26. Diagramas de barras que permiten observar la frecuencia de algunos de los compuestos mas usados en las capas de la celda solar de Perovskita. Donde (A) es el diagrama de barras de la capa *ETL*, (B) el diagrama de barras de la capa *HTL* y (C) el diagrama de barras de la capa *Perovskita*.

Analizados los resultados de las capas, se pasa a analizar los parámetros de desempeño, generando diagramas de dispersión donde se puede observar la tendencia de sus valores entre los años 2013 al 2018, se ven en la Figura 27 o en el Anexo 6. En los diagramas se observa la gran dispersión de los valores para los parámetros de las celdas solares de Perovskita, especialmente en los parámetros del PCE y Voc, siendo PCE el parámetro que mas los científicos desean que tome un valor mayor para mejorar la aplicación industrial de estas celdas solares.

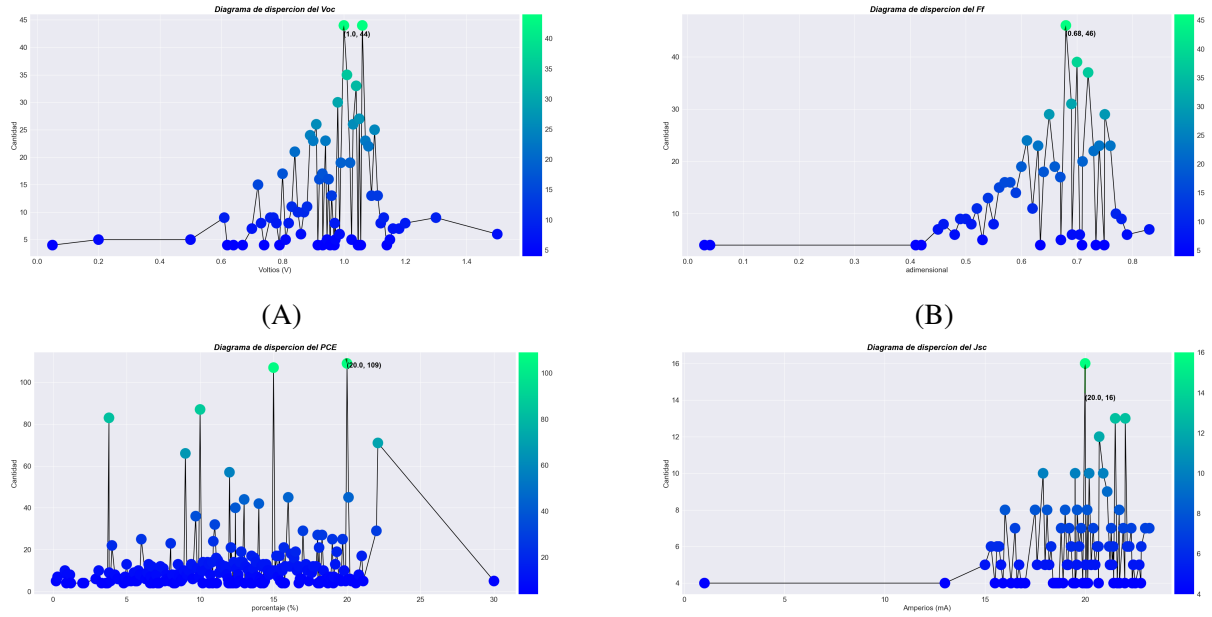

Figura 27. Diagramas que permiten observar la frecuencia para los valores de los parámetros de desempeño. Donde (A) son los diagramas del parámetro V_{oc}, (B) los diagramas del parámetro Ff, (C) los diagramas del parámetro PCE y (D) los diagramas del parámetro J_{sc}.

6. Recomendaciones

Al momento de escoger el tema sobre el cual se desea realizar la extracción automática de información o minería de texto, es necesario tener una noción completa de la nomenclatura y características especiales de su sintaxis. Esto con la idea de simplificar el trabajo de seleccionar la información necesaria para general los archivos de entrenamiento y de evaluación, generar las reglas de optimización y simplificar el trabajo de guardado.

7. Trabajos futuros

Actividades complementarias usadas o no como ayuda a los desarrollos presentados y que podrían ser mejoradas a futuro.

7.1. API en fase ALFA para la corrección de conjuntos de datos en formato .JSON para la creación de modelos NER de spaCy 3.0

Como se expresa en la sección 4.1.2.2 sobre la necesidad de corregir el conjunto de datos de entrenamiento y evaluación y evitar usar la herramienta de paga proporcionada por los desarrolladores de "*spaCy 3.0*", se decide crear la API en la sub-sub-carpeta "*GUI*" el 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, dos archivos PDFs llamados "$.Api\_about.pdf$" y "$.Api\_tutorial.pdf$" con información de la API y un tutorial para usar la API, respectivamente. Además se realiza un vídeo tutorial llamado "$Tutorial.mp4$".

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 28, permite ingresar el archivo .JSON estructurado como el del ejemplo de la Figura 16 de la sección 4.1.2.1 y un archivo nuevo o uno modificado en formato .JSON con la misma estructura como salida.

En la segunda ventana o la ventana principal que se ve en la Figura 29, 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 ultimo la ventana secundaria o tercera ventana que se ve en la Figura 30, se usa para buscar e ingresar la entidad nombrada y el nombre de entidad nombrada que no se reconoció automáticamente.

Figura 28. Primera ventana de la API, usada para ingresar el archivo con los datos a corregir y ingresar el archivo de salida a usar.

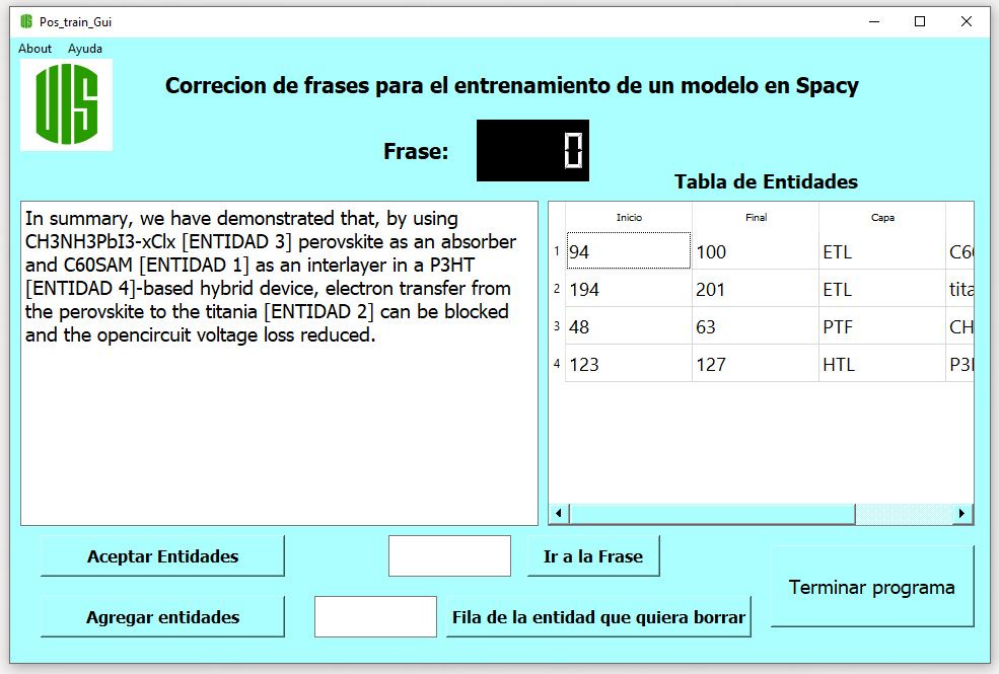

Figura 29. Ventana principal de la API, usada realizar la corrección visual y manual del archivo de datos.

Figura 30. Ventana secundaria de la API, usada para buscar la posición e ingresar las nuevas entidades nombradas.

7.2. Acercamiento a la extracción automática de tablas con información de parámetros de desempeño en artículos científicos en formato PDF

Durante el desarrollo del trabajo y en particular durante la sección 4.1.2 se observa la incidencia de entidades nombradas asociadas a los parámetros de desempeño de las celdas solares solo expresadas en las tablas y no en otras partes del texto del articulo. Esto hace que se decida incursionar de forma paralela en los métodos que se podrían usar para extraer además del texto de los PDFs, la información de sus tablas. Luego se genera un código sencillo que permite extraer al menos un pequeño porcentaje de todas las tablas presentes en los artículos y guardarlos en formato Excel (.xlsx).

Para el código se 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).

Se crea el scripts "*Table_extraction.py*" (ejemplo, Pseudocódigo 4) con el cual primero 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. Después de que se logre extraer alguna tabla, realiza un acondicionamiento para obtener su titulo y comprobar que efectivamente se trate de una, y no de un error causado por el formato de doble columna de los artículos científicos guardados en PDF. Por ultimo la tabla o tablas obtenidas son

guardadas en una hoja en un excel.

Pseudocódigo 4 Extracción de información exclusivamente de tablas presentes en los PDFs.

Require: PDF y nombre del excel para guardar las tablas.
Ensure: Archivo excel con las tablas encontradas.

function TABLE_EXTRACTION($pdf, nombre$)
    $estilos, Nfuentes \leftarrow$ FONTS(pdf) ▷ Segmentation.py, estilos y "N"fuentes del pdf.
    $etiqueta \leftarrow$ FONT_TAGS($estilos, Nfuentes$) ▷ Segmentation.py, etiquetado de las partes del pdf.
    $bloques \leftarrow$ HEADERS($pdf, etiqueta$) ▷ Segmentation.py, lista con los bloques del pdf.
    for la cantidad de *bloques* **do**
        $titulo \leftarrow$ usa expresiones regulares para buscar el numero de tabla en *bloques*.
    end for
    for *pagina* en el pdf **do**
        tabla ← usa expresiones regulares para buscar si hay una tabla en *pagina*.
        if existe *tabla* **then**
            Data ← TABULA(*tabla*, pdf) usar tabula para extraer la tabla en la posición *tabla*.
            DataC ← adaptación, corrección de *Data* y asociación con $titulo$
            Excel ← Guarda *DataC* en un excel con el nombre *nombre*
        else
            Data ← TABULA(*pagina*, pdf) usar tabula para extraer la tabla en toda la *pagina*.
            DataC ← adaptación, corrección de *Data* y asociación con $titulo$
            Excel ← Guarda *DataC* en un excel con el nombre *nombre*
        end if
    end for

8. Conclusiones

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

- Se ha logrado implementar una herramienta de software para la detección y localización de entidades nombradas relacionadas con los parámetros desempeño de las celdas solares mediante la aplicación de Procesamiento de Lenguaje Natural. También la herramienta se amplio para funcionar con las entidades nombradas asociadas a las capas de Perovskita. Con la herramienta se esperaría ayudar a automatizar la extracción de grandes cantidades de datos de celdas solares convencionales de perovskita contenidas en cantidades cada vez mayores de artículos científicos en formato PDF.

- El proceso de limpieza, decodificación y estandarización de los archivos de PDF, se realizo de manera semi-manual a raíz del conocimiento necesario para comprender los compuestos que conforman las capas y los parámetros de desempeño de las celdas solares Perovskita. Este trabajo podría hacerse mas sencillo si las revistas científicas normalizaran la codificación con las que publican sus artículos y si los artículos científicos sobre celdas solares estandarizaran su nomenclatura especifica al tema de la química orgánica.

- En torno a los trabajos con planteamientos parecidos en la web, que usan la librería spaCy, este proyecto creo y aplico un modelo NER mediante un método flexible que permite su modificación, entrenamiento y uso para aplicaciones diferentes a la planteada en el proyecto.

- El modelo NER desarrollado en el presente trabajo muestra desempeños por arriba del 90 %, lo cual está en línea con literatura reciente donde se aplica NER para problemas similares pero diferentes a celdas solares de Perovskita.

- La herramienta de software del proyecto se estructuro y guardo para acercarse un poco a la utilidad que tendría una librería de Python consolidada, permitiendo realizar casi todo el proceso descrito en el proyecto a través de comandos sencillos en la consola de comandos (cmd).

Referencias Bibliográficas

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

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

Ariga, A. (2021). 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).

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/.

Clark, E. G. (2021). Openpyxl, https://openpyxl.readthedocs.io/en/stable/index.html.

DS New Energy (2019). Una introducción a las células solares perovskitas y perovskitas, https://www.dsisolar.com/info/an-introduction-to-perovskites-and-perovskite-36067671.html.

Eisenstein, J. (2018). Natural language processing.

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.

Gencay, R. and Qi, M. (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks*, 12(4):726–734.

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

Graetzel, M., Janssen, R. A., Mitzi, D. B., and Sargent, E. H. (2012). Materials interface engineering for solution-processed photovoltaics. *Nature*, 488(7411):304–312.

GraphEverywhere (2022). ¿qué es el clustering?, https://www.grapheverywhere.com/que-es-el-clustering/.

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(17).

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.

Honnibal, M. and Montani, I. (2015). https://spacy.io.

Honnibal, M. and Montani, I. (2021a). AVANCED NLP with spaCy, https://course.spacy.io/es.

Honnibal, M. and Montani, I. (2021b). prodigy. radically efficient machine teaching. an annotation tool powered by active learning., https://prodi.gy/.

Honnibal, M. and Montani, I. (2021c). Rule-based Matcher Explorer, https://explosion.ai/demos/matcher.

Honnibal, M. and Montani, I. (2021d). spaCy 101: Everything you need to know, https://spacy.io/usage/spacy-101.

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.

Izaurieta, F. and Saavedra, C. (2000). Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*.

Jessop, D. M., Adams, S. E., Willighagen, E. L., Hawizy, L., and Murray-Rust, P. (2011). Chemicaltagger: A tool for semantic text-mining in chemistry. *J Cheminform*, 3(1).

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.

Kim, M., Seo, J., Lu, H., Ahlawat, P., Mishra, A., Yang, Y., Hope, M. A., Eickemeyer, F. T., Kim, M., Yoon, Y. J., Choi, I. W., Darwich, B. P., Choi, S. J., Jo, Y., Lee, J. H., Walker, B., Zakeeruddin, S. M., Emsley, L., Rothlisberger, U., Hagfeldt, A., Kim, D. S., Grätzel, M., and Kim, J. Y. (2018). Perovskites photovoltaic solar cells: An overview of current status. *Renewable and Sustainable Energy Reviews*, 91:1025–1044.

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.

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

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

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.

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.

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

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.

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

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

NREL (2022). Photovoltaic research , `https://www.nrel.gov/pv/cell-efficiency.html`.

Odabasi, C. and Yildirim, R. (2020). Assessment of reproducibility, hysteresis, and stability relations in perovskite solar cells using machine learning. *Energy Technology*, 8(12):1901449.

Olivetti, E. A., Cole, J. M., Kim, E., Kononova, O., Ceder, G., Han, T. Y.-J., and Hiszpanski, A. M. (2020). Data-driven materials research enabled by natural language processing and information extraction. *Applied Physics Reviews*, 7(4):041317.

Padró, L. and Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. In *LREC2012*, Istanbul, Turkey. ELRA.

Pai, A. (2020). What is tokenization in nlp? here's all you need to know, https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/.

Parikh, N., Karamta, M., Yadav, N., Mahdi Tavakoli, M., Prochowicz, D., Akin, S., Kalam, A., Satapathi, S., and Yadav, P. (2022). Is machine learning redefining the perovskite solar cells? *Journal of Energy Chemistry*, 66:74–90.

Paszke, A., Gross, S., Chintala, S., and Chanan, G. (2017). Pytorch, https://pytorch.org/.

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.

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

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

Python Software Foundation (2022). The python standard library, https://docs.python.org/3/library/index.html.

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.

Ross, M. A. J. (2017). Fabricacion y caracterización de celdas solares orgánicas basadas en perovskita.

S. REN21 (2021a). RENEWABLES 2021, GLOBAL STATUS REPORT. A COMPREHENSIVE ANNUAL OVERVIEW OF THE STATE OF RENEWABLE ENERGY, https://www.ren21.net/gsr-2021/.

S. REN21 (2021b). Why is renewable energy important?, https://www.ren21.net/why-is-renewable-energy-important/.

Salas, R. (2004). Redes neuronales artificiales. *Universidad de Valparaıso. Departamento de Computación*, 1:1–7.

Shi, Z. and Jayatissa, A. H. (2018). Perovskites-based solar cells: a review of recent progress, materials and processing methods. *Materials*, 11(5):729.

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. PMID: 27669338.

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.

Vidal, R., Alberola-Borràs, J.-A., Sánchez-Pantoja, N., and Mora-Seró, I. (2021). Comparison of perovskite solar cells with other photovoltaics technologies from the point of view of life cycle assessment. *Advanced Energy and Sustainability Research*, 2(5):2000088.

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

Apéndices

Apéndice A. Lista de Pseudocódigos

Pseudocódigo 1 Entrenamiento del modelo.

Require: Numero de ciclos, numero de lotes, la tasa de olvido y el archivo .JSON de entrenamiento.
Ensure: Modelo entrenado con la librería spaCy 3.0.

```
1:  function TRAIN(frases, ciclos)
2:      modelo ← nuevo modelo en blanco
3:      spacy ← modelo base proporcionado por spaCy 3.0
4:      modelo.ner ← cargar el pipeline NER para el modelo en blanco desde spaCy3.0
5:      modelo.ner.entidades ← inicializar todas las nuevas entidades nombradas
6:      for 0:ciclos do
7:          Reorganizar de forma aleatoria el orden de los elementos del archivo frases.
8:          Crea grupos de elementos de tamaño lotes de los elementos reorganizados.
9:          for cada lote en lotes do
10:             for Cada frase y sus entidades nombradas en el grupo lote do
11:                 F ← Frase tomada
12:                 E ← lista de entidades nombradas tomada
13:                 Predice las entidades nombradas de la frase y las compara con las ingresadas.
14:                 P ← lista de entidades nombradas predichas
15:                 Aplica la tasa de olvido (drop).
16:                 Calcula las perdidas del ciclo de entrenamiento (losses).
17:                 Actualiza el modelo.
18:             end for
19:         end for
20:     end for
21:     modelo.pipelines ← cargar los demás pipeline para el modelo
22:     modelo ← Guarda los cambios en el modelo modelo
```

Pseudocódigo 2 Evaluar el modelo usando spaCy 3.0.

Require: Archivo con oraciones y/o texto con sus entidades nombradas reconocidas.
Ensure: Archivo excel con la evaluación del modelo.

```
1: function SCORE(Modelo, Frases, Archivo)
2:     oraciones ← Oraciones y/o texto de Frases
3:     EnV ← Entidades nombradas verdaderas de las oraciones y/o texto de Frases
4:     for cada oracion en oraciones do
5:         Ent ← Aplica el modelo Modelo a oracion y extrae las entidades nombradas
6:         Precision, recall, fcore ← Aplica las 3 métricas a oracion usando spaCy 3.0, EnV y Ent
7:     end for
8:     Excel ← Guarda Precision, recall, fcore en un excel con el nombre Archivo
```

Pseudocódigo 3 Corrección y optimización de la extracción.

Require: Modelo entrenado y el PDF a extraer información.
Ensure: Diccionario con la información de las capas, los parámetros y las oraciones extraídas.

```
1: function TEXT_EXTRACTION(Modelo, PDF)
2:     secciones ← SEGMENTATION(PDF) ▷ Llama el procedimiento de sub-sección 4.4.1.
3:     for cada seccion en secciones do
4:         Texto ← CLEAN(seccion)          ▷ Limpia, codifica y estandariza, sub-seccion 4.1.1.
5:         for Cada oracion en Texto do
6:             Entidad1 ← Entidades nombradas asociadas a los parámetros, con Modelo
7:             Entidad2 ← Entidades nombradas asociadas a los valores de los parámetros, con Modelo
8:             Entidad3 ← Entidades nombradas asociadas a las capas, con Modelo
9:             Diccionario ← Guarda entidad3 en el diccionario
10:            if entidad1 existe en oracion then
11:                if entidad2 es un numero y es mayor a 0.0 then
12:                    Diccionario ← Guarda entidad2 en el diccionario
13:                end if
14:            end if
15:            Diccionario ← Guarda oracion en el diccionario
16:        end for
17:    end for
```

Pseudocódigo 4 Extracción de información exclusivamente de tablas presentes en los PDFs.

Require: PDF y nombre del excel para guardar las tablas.
Ensure: Archivo excel con las tablas encontradas.

```
1: function TABLE_EXTRACTION(pdf, nombre)
2:     estilos, Nfuentes ← FONTS(pdf)          ▷ Segmentation.py, estilos y "N"fuentes del pdf.
3:     etiqueta ← FONT_TAGS(estilos, Nfuentes)  ▷ Segmentation.py, etiquetado de las partes del pdf.
4:     bloques ← HEADERS(pdf, etiqueta)         ▷ Segmentation.py, lista con los bloques del pdf.
5:     for la cantidad de bloques do
6:         titulo ← usa expresiones regulares para buscar el numero de tabla en bloques.
7:     end for
8:     for pagina en el pdf do
9:         tabla ← usa expresiones regulares para buscar si hay una tabla en pagina.
10:        if existe tabla then
11:            Data ← TABULA(tabla, pdf) usar tabula para extraer la tabla en la posición tabla.
12:            DataC ← adaptación, corrección de Data y asociación con titulo
13:            Excel ← Guarda DataC en un excel con el nombre nombre
14:        else
15:            Data ← TABULA(pagina, pdf) usar tabula para extraer la tabla en toda la pagina.
16:            DataC ← adaptación, corrección de Data y asociación con titulo
17:            Excel ← Guarda DataC en un excel con el nombre nombre
18:        end if
19:    end for
```

Apéndice B. Resumen del esquema técnico

En el presente anexo se desarrolla un resumen de la infraestructura técnica usada, el repositorio de los datos y la estructura de la herramienta de software desarrollada para la implementaron de la tesis guardada en la carpeta "*NerPSC*".

Repositorio del proyecto en Github

Este proyecto utiliza la capa gratuita del repositorio Github para almacenar la herramienta de software con formada por: este documento de proyecto de grado (*Tesis_Mary_HERRAMIENTA DE SOFTWARE PARA LA EXTRACCIÓN AUTOMÁTICA....pdf*), las carpetas "*NerPSC*" y "*Imagenes*", y los archivos "*Instrucciones-install.txt*", "*requirements.txt*" y "*README.md*". Github es una plataforma de desarrollo colaborativo con control de versiones muy usada para el almacenamiento de proyectos.

La URL del repositorio es la siguiente: TESIS-NerPSC

Estructura de la herramienta de software

En la herramienta de software creada en el repositorio "*TESIS-NerPSC*" se encuentra la carpeta "*NerPSC*", Figura 31, donde se encuentran la sub-carpeta "*data_pdf*", los archivos .txt, .json, .png y .xlsx de resultados tanto fuera como dentro de la sub-carpeta "*DATA*", el script "*Module.py*" que funciona como interfaz de línea de comandos del proyecto, la carpeta del modelo entrenado "*Model_NerPSC*", las carpetas de los modelos de validación cruzada, la sub-carpeta "*Modulo_NerPSC*" con todos los demás scripts, Figura 32 y el script "*Process.py*" que importa las principales funciones. Todos los pasos del proyecto pueden ser llamados y usados con los scripts "*Process.py*" y "*Module.py*".

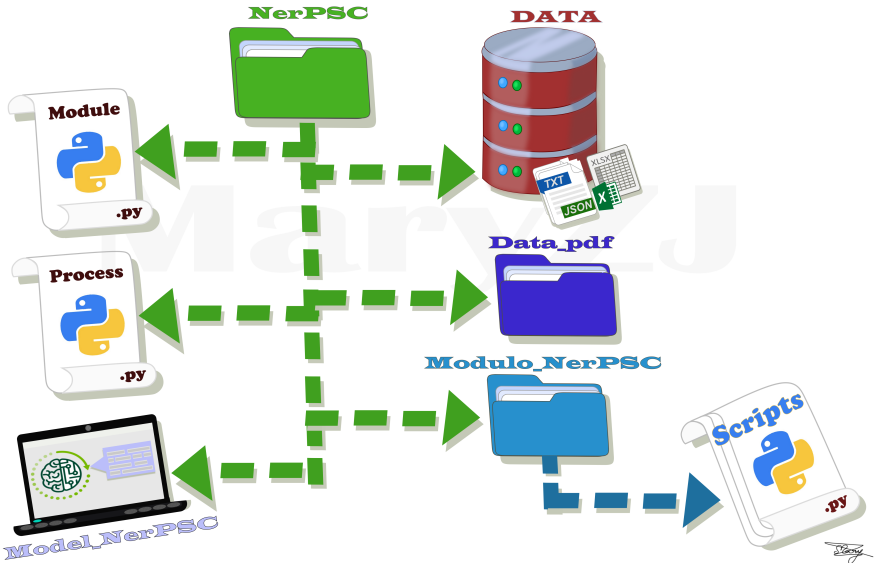

Figura 31. Diagrama de la estructura de la carpeta principal del proyecto ("*NerPSC*").

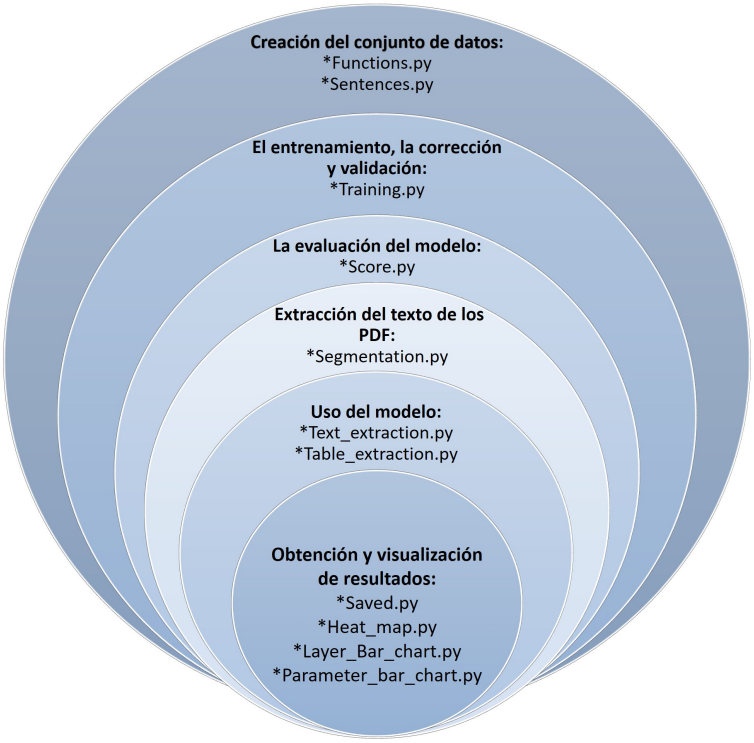

Figura 32. Diagrama de la estructura de la sub-carpeta con los scripts usados para el proyecto ("*Modulo_NerPSC*").

Infraestructura técnica

En el proyecto todos los scripts usados se elaboraron con el lenguaje de programación **Python versión 3.8** y la aplicación **Anaconda**. Se utilizaron algunas librerías de licencia libre tanto propias de python (Python Software Foundation, 2022) como creadas por autores independientes . El resumen de las librerías usadas es:

- **Os:** Sirve para administrar fácilmente las rutas de ficheros locales y manipular la estructura de directorios (para leer y escribir archivos.
- **Sys:** Es el encargado de proveer variables y funcionalidades, directamente relacionadas con el intérprete.
- **Re:** Módulo proporciona operaciones de coincidencia de expresiones regulares.
- **Json:** formato de intercambio de datos liviano inspirado en la sintaxis literal de objetos de JavaScript.
- **Pandas** herramienta de manipulación y análisis de datos de código abierto, construida sobre el lenguaje de programación Python. Autores: 2000 colaboradores voluntarios con Wes McKinney como director(McKinney, 2022).
- **Openpyxl:** Biblioteca de Python para leer/escribir archivos Excel. Autor: Eric GazoniCharlie Clark (Clark, 2021).
- **Datetime:** Módulo que proporciona clases para manipular fechas y horas.

- **Tabula-py:** Contenedor Python simple de tabula-java , que puede leer tablas de PDF. Autor: Aki Ariga (Ariga, 2021).

- **Random:** Módulo que implementa generadores de números pseudoaleatorios para varias distribuciones.

- **Operator:** Exporta un conjunto de funciones eficientes correspondientes a los operadores intrínsecos de Python.

- **spaCy:** Librería de uso industrial para el procesamiento del lenguaje natural.Autores: Matthew Honnibal, Ines Montani y Justin DuJardin et al (Honnibal and Montani, 2015).

- **Qt-designer:** Herramienta para crear rápidamente interfaces gráficas de usuario con widgets del marco Qt GUI . Le brinda una interfaz simple de arrastrar y soltar para diseñar componentes como botones, campos de texto, cuadros combinados y más. Autores: Qt Group (Nasdaq Helsinki: QTCOM) (Herrmann, 2021).

- **PyQt5:** Conjunto completo de enlaces de Python para Qt v5. Se implementa como más de 35 módulos de extensión y permite que Python se use como un lenguaje de desarrollo de aplicaciones alternativo a C++ en todas las plataformas compatibles, incluidos iOS y Android. Autores: Riverbank Computing Limited (Riverbank Computing Limited, The Qt Company, 2021).

Apéndice C. Tabla y gráficas con la información del proceso de entrenamiento, corrección, validación y evaluación de 19 modelos.

Tabla 7

Tabla con la compilación de los modelos entrenados, con sus parámetros resultado de la evaluación y el tiempo de entrenamiento, guardada en el archivo "Table7.xlsx".

Modelo	Metricas	precisión	recall	f-score	N	B	D	Tiempo
0	**mean**	0,940936	0,946596	0,938211	100	1	0.5	10:46:08.937
	std	0,110481	0,108298	0,099410	100	1	0.5	10:46:08.937
2	**mean**	0,88343491	0,905997	0,886833	100	2	0.1	5:58:37.391
	std	0,19324828	0,176933	0,175135	100	2	0.1	5:58:37.391
1	**mean**	0,91970914	0,94709	0,926015	100	10	0.5	2:39:47.6751
	std	0,13309155	0,108215	0,111905	100	10	0.5	2:39:47.6751
3	**mean**	0,93263331	0,942927	0,934126	30	1	0.2	3:23:09.695
	std	0,14034465	0,137894	0,132799	30	1	0.2	3:23:09.695
4	**mean**	0,92821432	0,92987	0,923224	40	1	0.2	4:33:45.528
	std	0,15365642	0,152632	0,145326	40	1	0.2	4:33:45.528
5	**mean**	0,8626674	0,840948	0,846333	20	1	0.15	2:13:52.939
	std	0,22734874	0,238103	0,226871	20	1	0.15	2:13:52.939
6	**mean**	0,92737575	0,933395	0,926513	35	1	0.2	3:53:48.481
	std	0,14430043	0,143332	0,13518	35	1	0.2	3:53:48.481
7	**mean**	0,86601394	0,894514	0,87644	50	1	0.5	5:47:31.291
	std	0,1805524	0,167831	0,169222	50	1	0.5	5:47:31.291
8	**mean**	0,92179271	0,951713	0,932727	30	1	0.2	3:24:22.552
	std	0,15326006	0,130788	0,135314	30	1	0.2	3:24:22.552
9	**mean**	0,92417233	0,945586	0,928242	30	1	0.2	3:15:31.746
	std	0,12219312	0,117111	0,106968	30	1	0.2	3:15:31.746
10	**mean**	0,92976117	0,937114	0,924718	30	10	0.2	0:46:23.958
	std	0,1254602	0,134311	0,118913	30	10	0.2	0:46:23.958
11	**mean**	0,93027737	0,940173	0,929685	100	10	0.2	2:34:12.466
	std	0,11395993	0,122072	0,1085	100	10	0.2	2:34:12.466
12	**mean**	0,93344992	0,932474	0,925467	85	10	0.2	2:10:55.458
	std	0,12558383	0,144015	0,125815	85	10	0.2	2:10:55.458
13	**mean**	0,93001751	0,93219	0,925939	85	5	0.2	3:14:25.493
	std	0,11641041	0,130203	0,113861	85	5	0.2	3:14:25.493
14	**mean**	0,93344992	0,932474	0,925467	65	100	0.2	0:49:35.791
	std	0,12558383	0,144015	0,125815	65	100	0.2	0:49:35.791
15	**mean**	0,93008145	0,946706	0,93253	100	100	0.2	1:17:58.669
	std	0,12324879	0,122325	0,113272	100	100	0.2	1:17:58.669
16	**mean**	0,9398309	0,9416669	0,9355227	85	100	0.2	1:10:39.854
	std	0,12099043	0,12355039	0,1137278	85	100	0.2	1:10:39.854
17	**mean**	0,93691693	0,942057	0,931654	200	100	0.2	2:20:27.205
	std	0,11865036	0,11582	0,104015	200	100	0.2	2:20:27.205
18	**mean**	0,92592519	0,949932	0,932014	400	100	0.2	4:36:38.747
	std	0,1281841	0,11727	0,113067	400	100	0.2	4:36:38.747

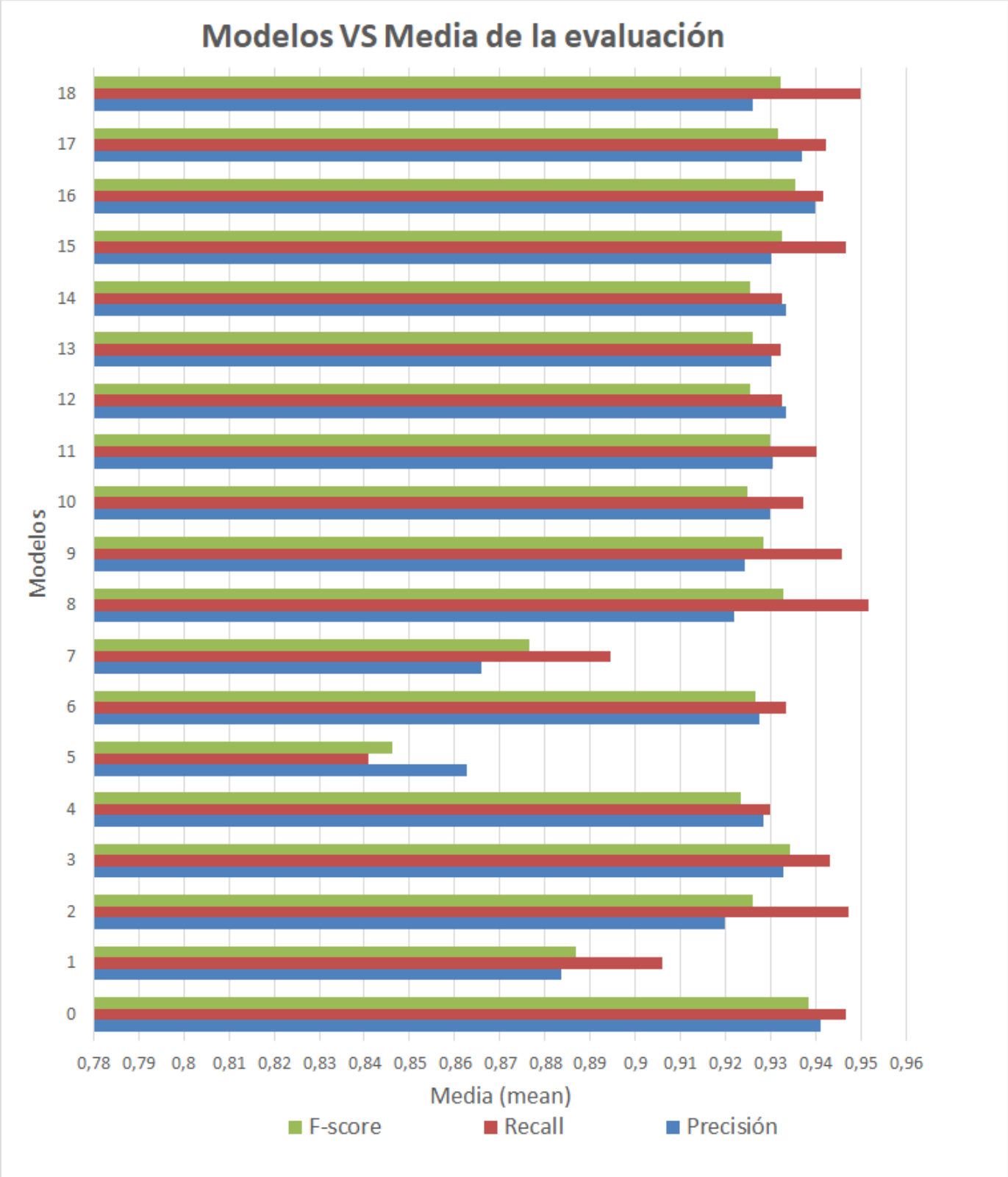

Figura 33. Diagrama de barras de los valores medios(mean) de evaluación de los 19 modelos entrenados en la sub-sección 5.1.2.

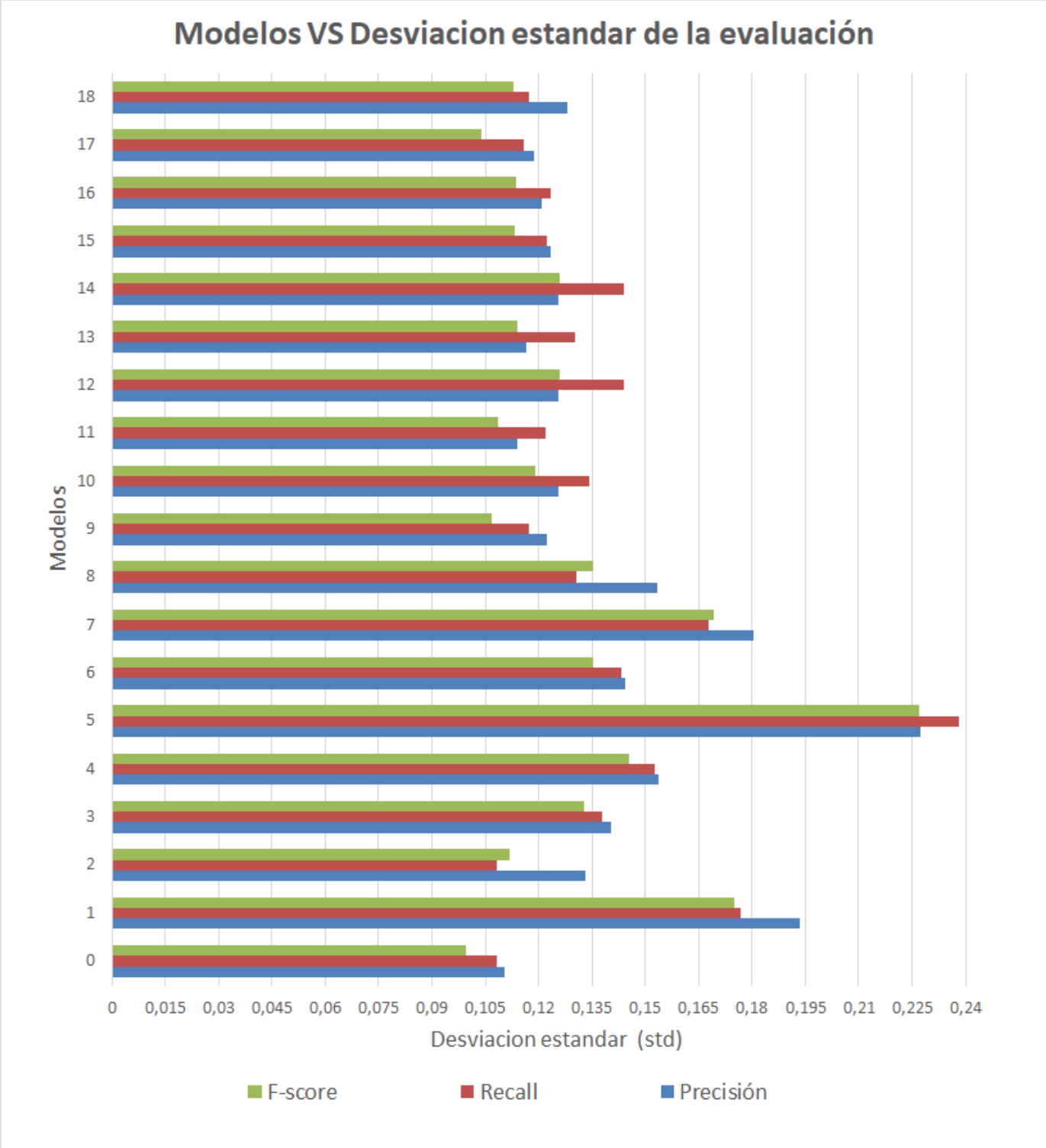

Figura 34. Diagrama de barras de los valores de desviación estándar(std) de evaluación de los 19 modelos entrenados en la sub-sección 5.1.2.

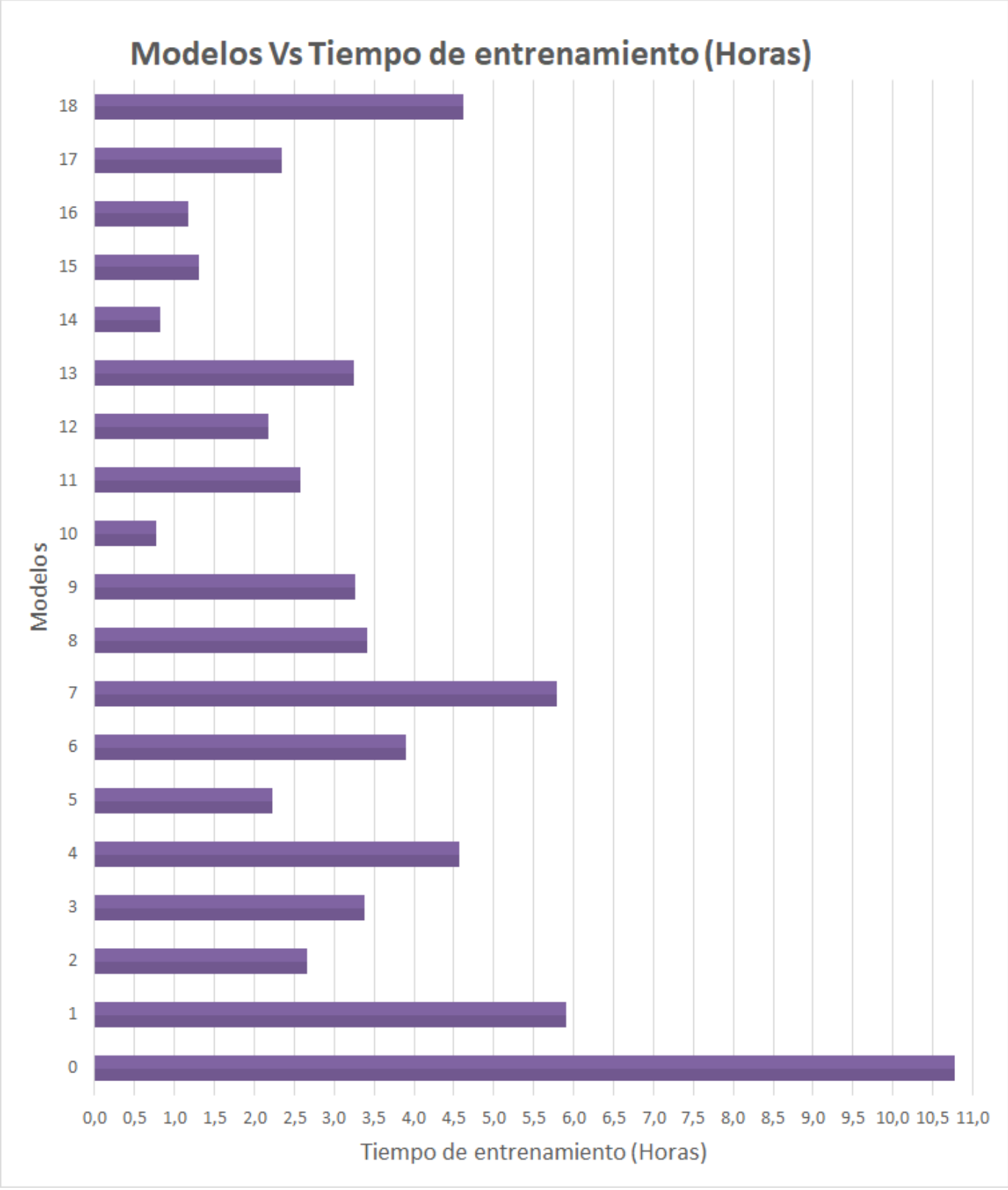

Figura 35. Diagrama de barras del tiempo de entrenamiento en horas de los 19 modelos entrenados en la sub-seccion 5.1.2.

Apéndice D. Diagramas de Calor para las capas de las celdas solares de Perovskita en el análisis de resultados.

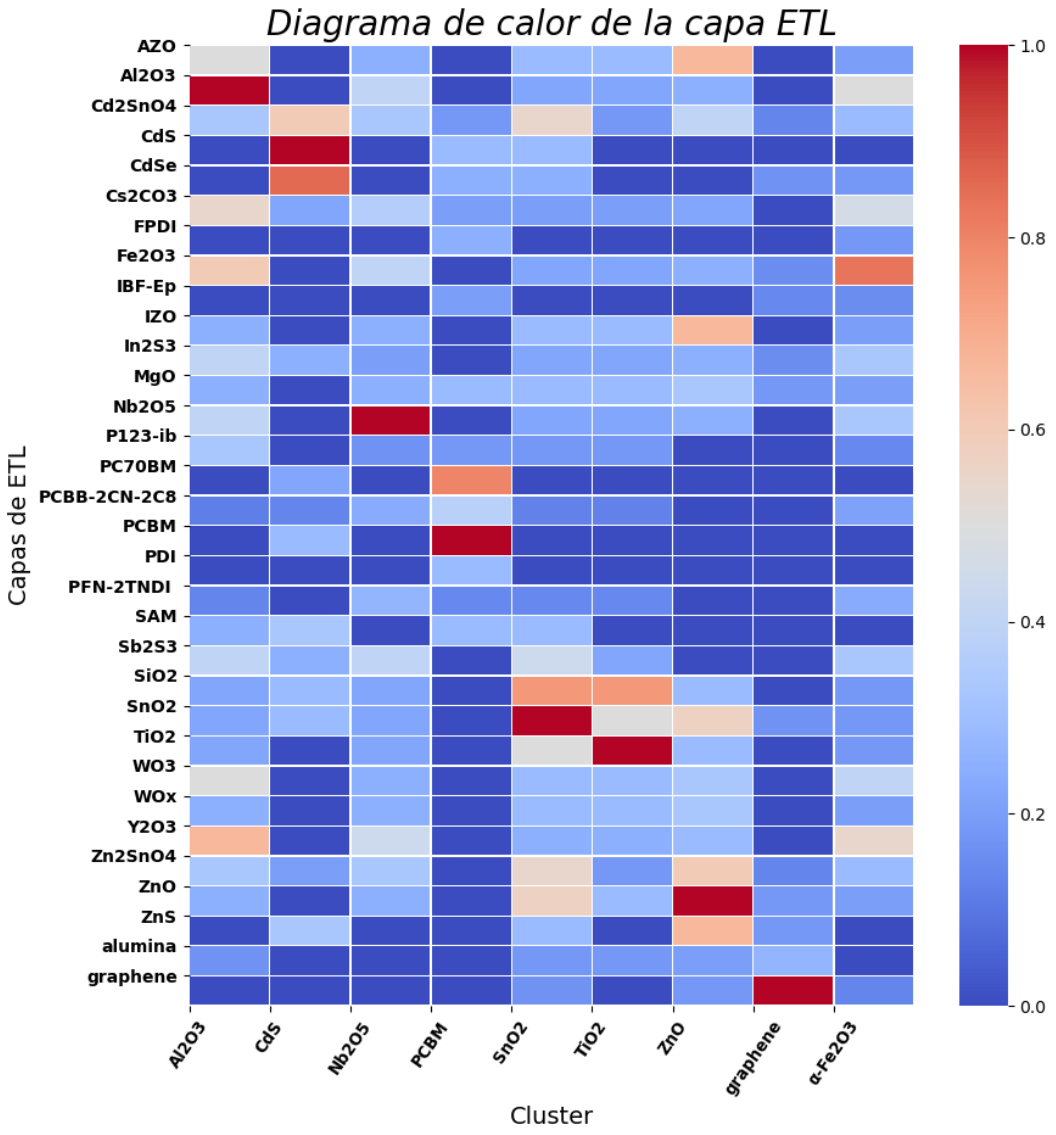

Figura 36. Diagrama de calor de los compuestos usados en la capa ETL.

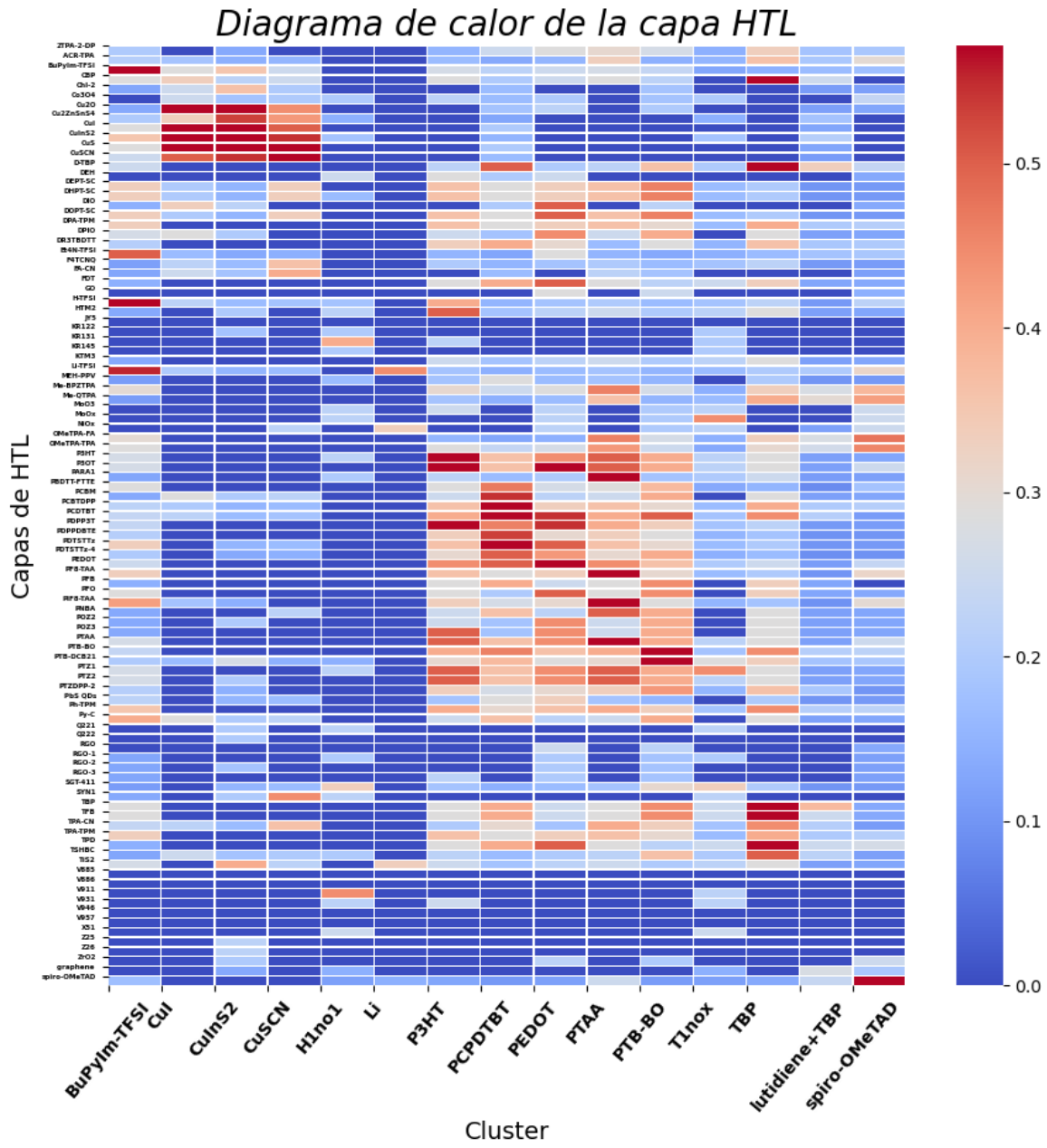

Figura 37. Diagrama de calor de los compuestos usados en la capa HTL.

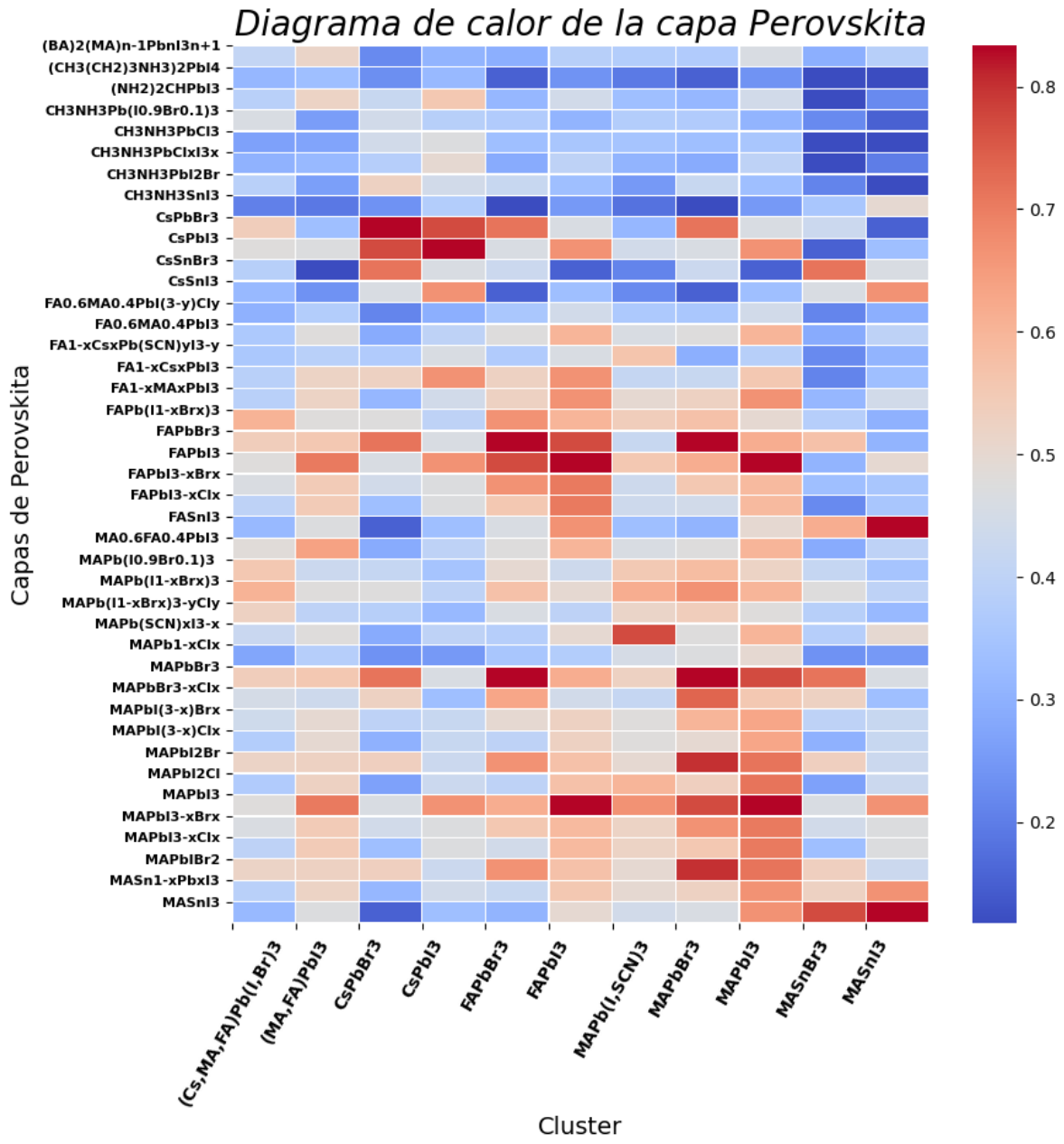

Figura 38. Diagrama de calor de los compuestos usados en la capa Perovskita.

Apéndice E. Diagramas de Barras para las capas de las celdas solares de Perovskita en el análisis de resultados.

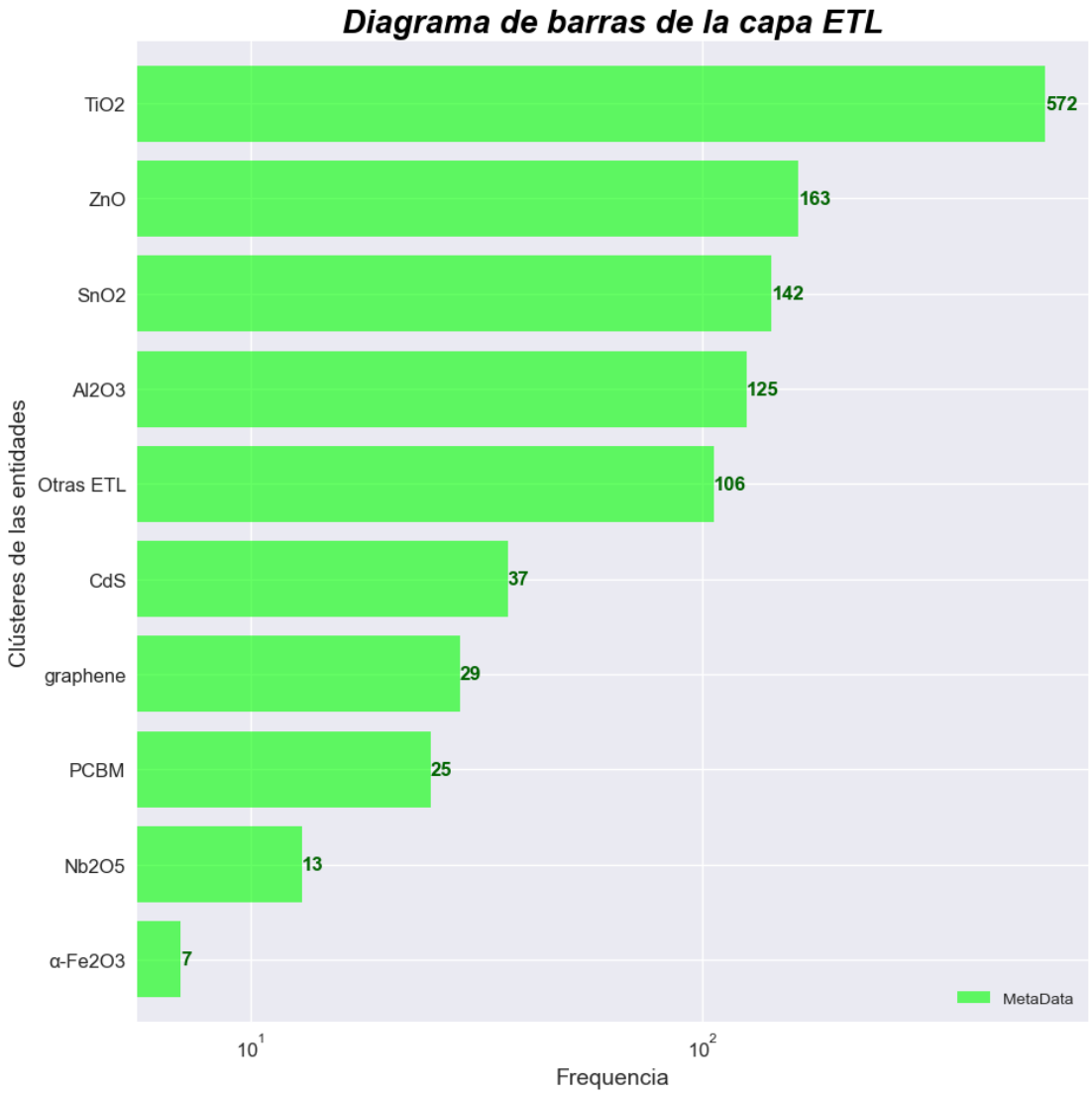

Figura 39. Diagrama de barras que permite observar la frecuencia de algunos de los compuestos mas usados en la capa ETL.

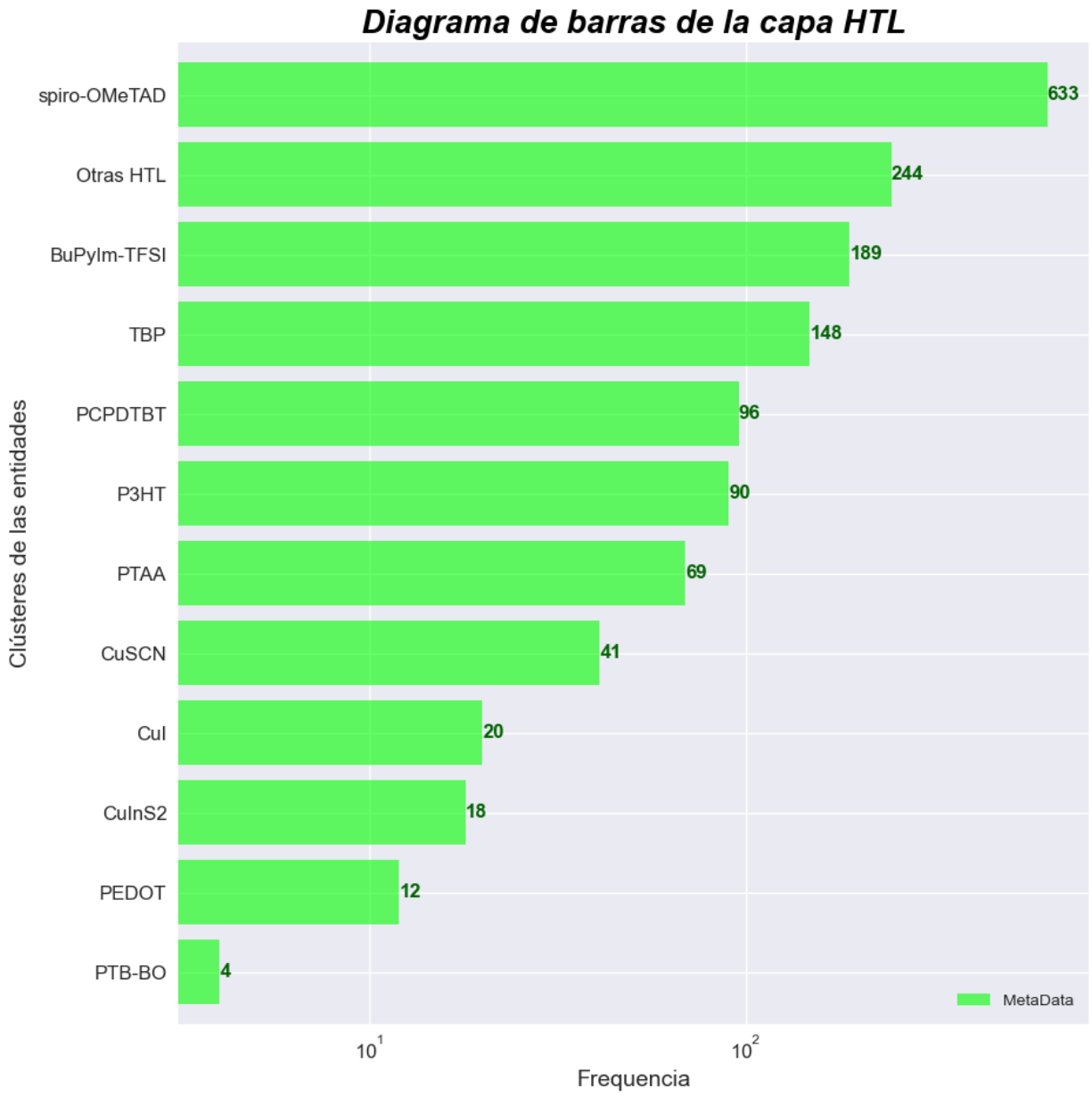

Figura 40. Diagrama de barras que permite observar la frecuencia de algunos de los compuestos mas usados en la capa HTL.

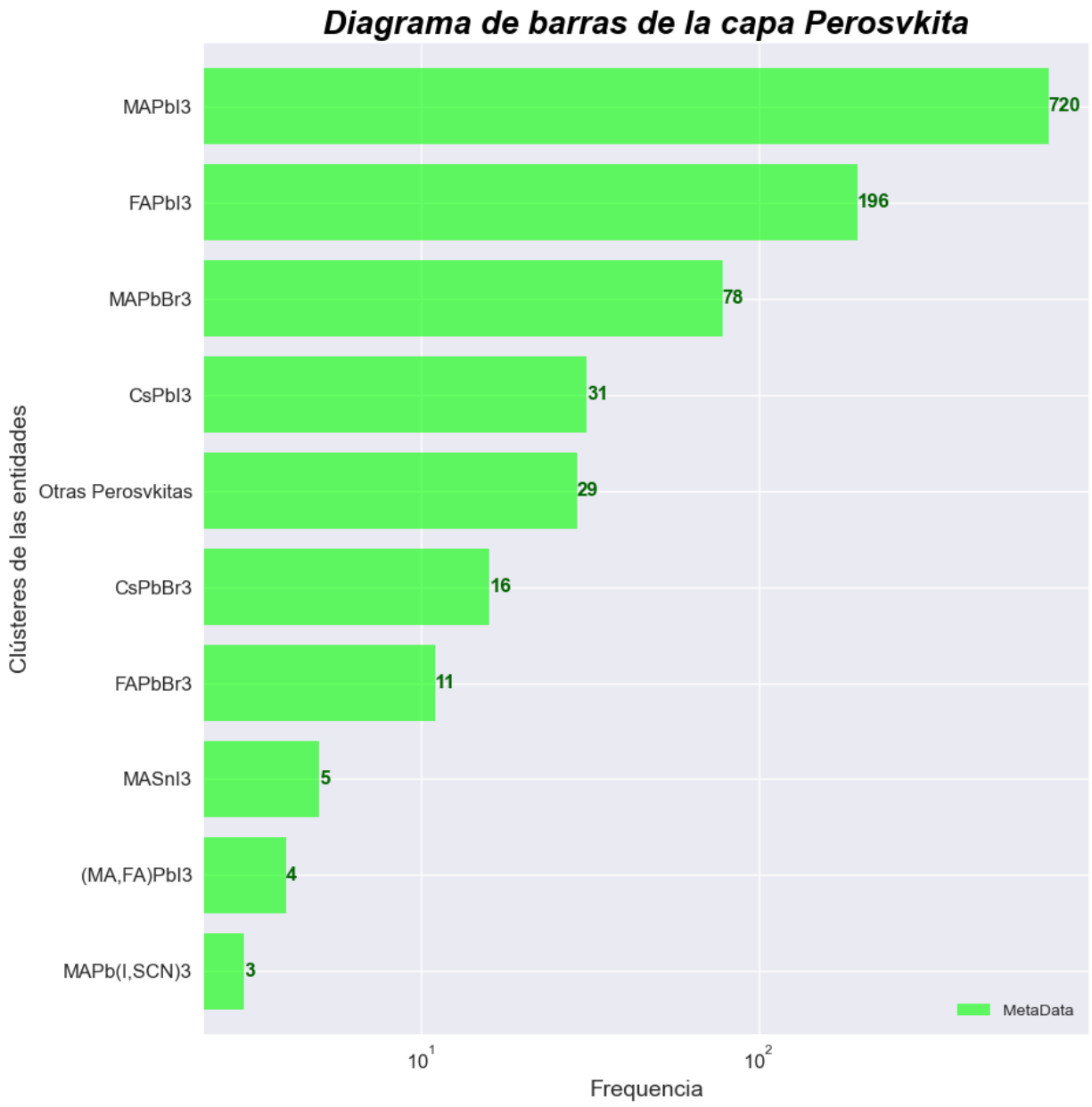

Figura 41. Diagrama de barras que permite observar la frecuencia de algunos de los compuestos mas usados en la capa Perovskita.

Apéndice F. Diagramas de Dispersión para los parámetros característicos de las celdas solares de Perovskita en el análisis de resultados.

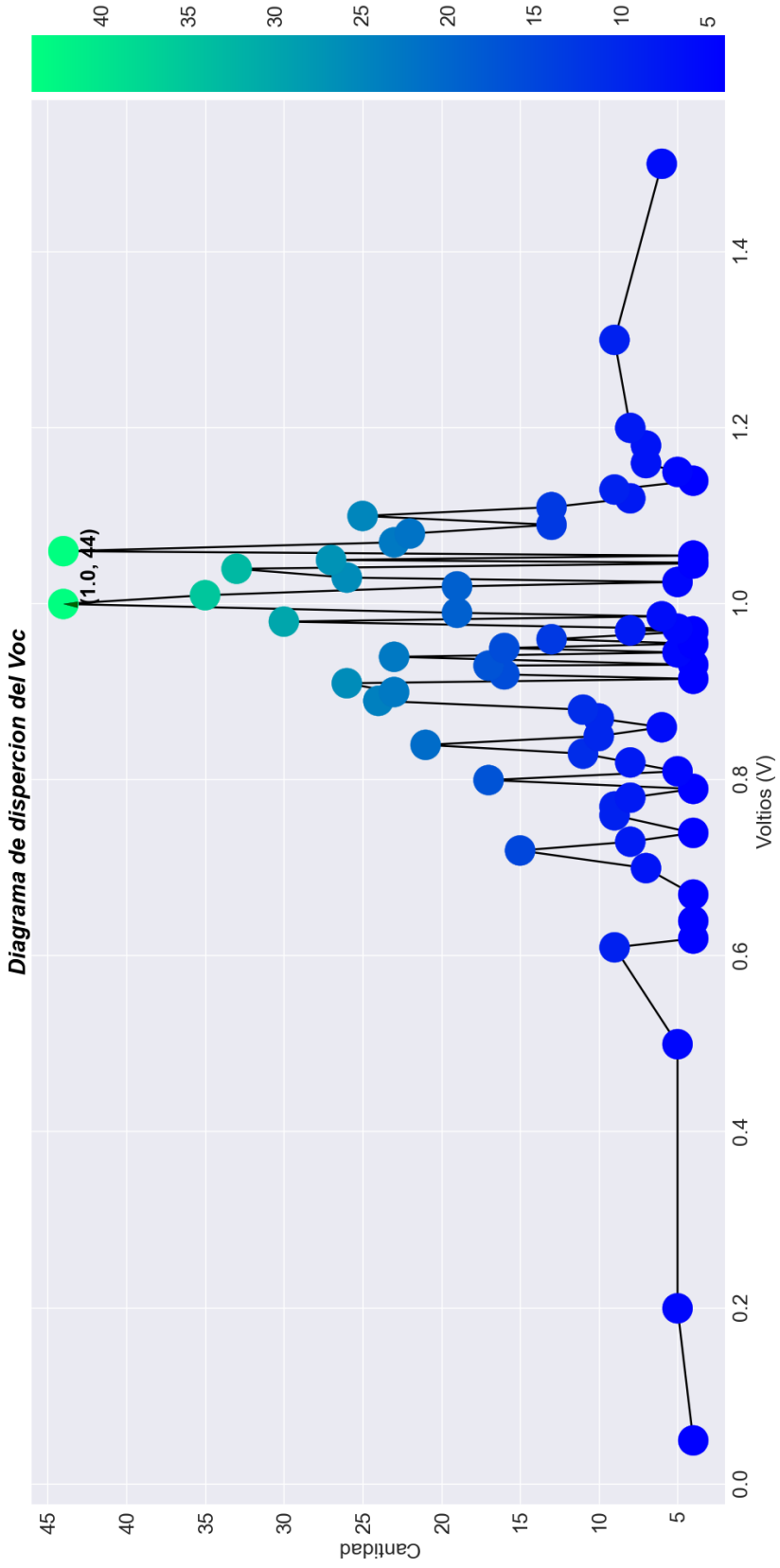

Figura 42. Diagrama de dispersión que permiten observar la frecuencia para los valores del parámetro VOC (Voltaje de circuito abierto).

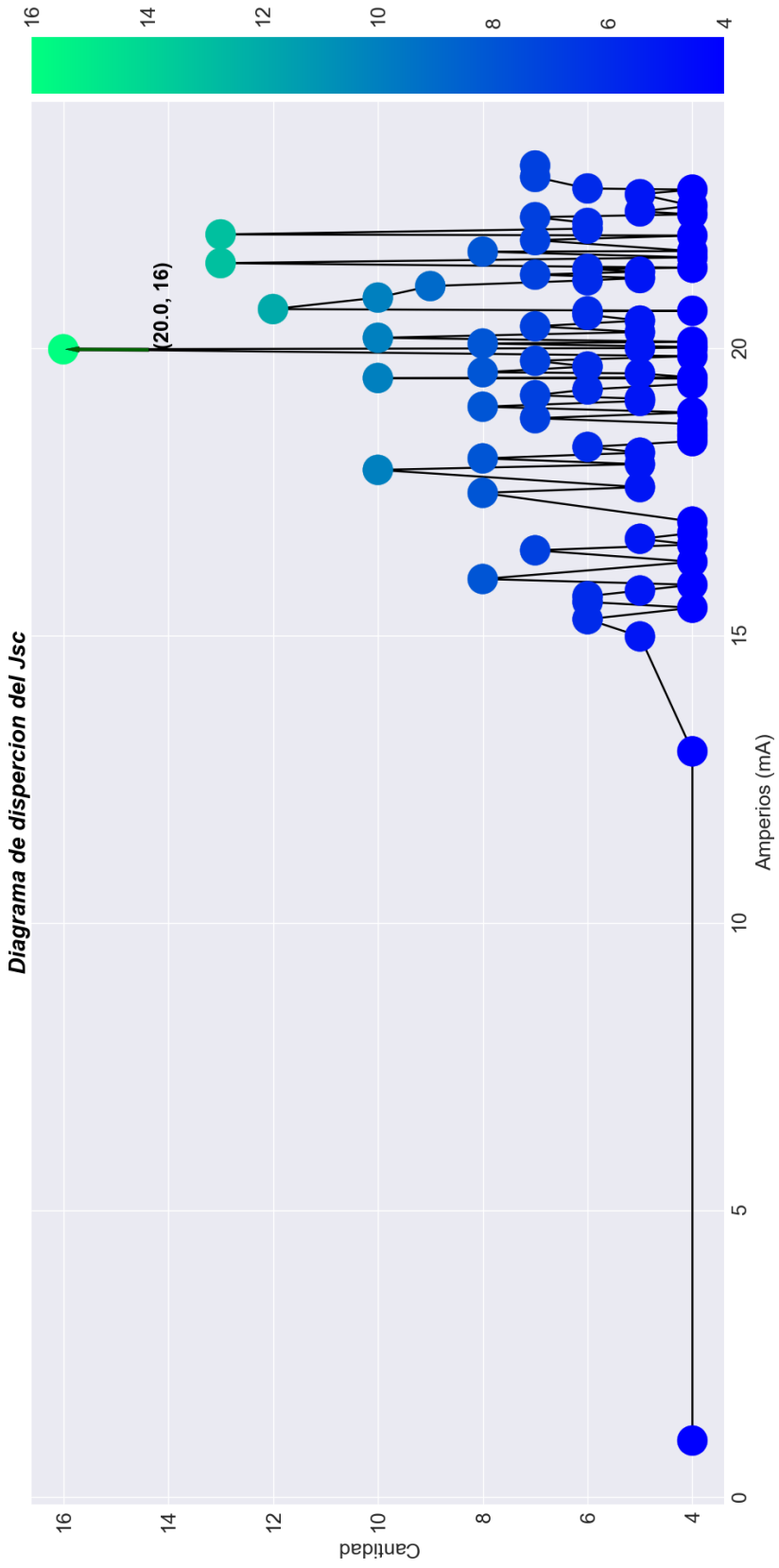

Figura 43. Diagrama de dispersión que permiten observar la frecuencia para los valores del parámetro JSC (Corriente de corto circuito).

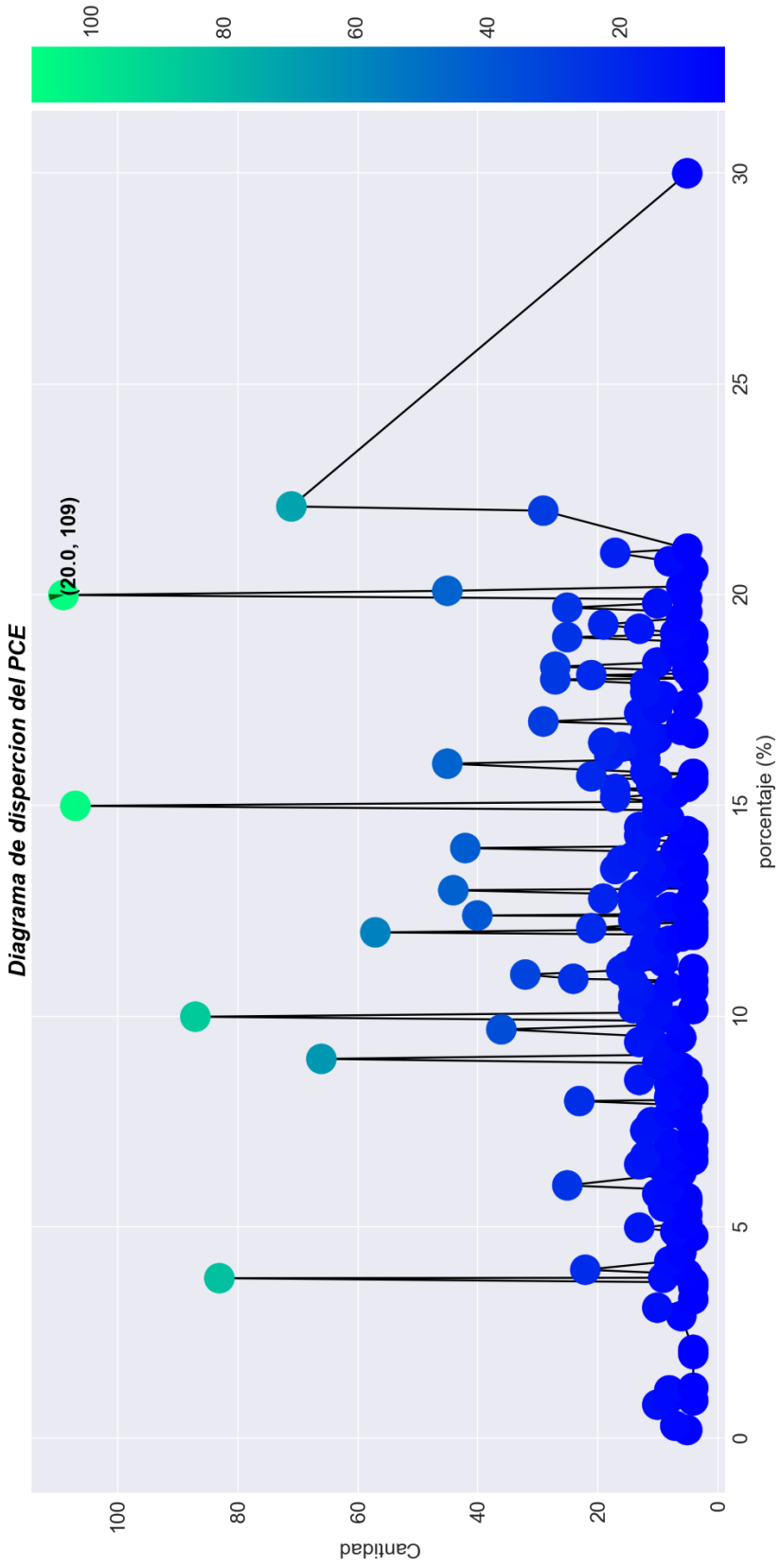

Figura 44. Diagrama de dispersión que permiten observar la frecuencia para los valores del parámetro PCE (eficiencia de conversión de Potencia).

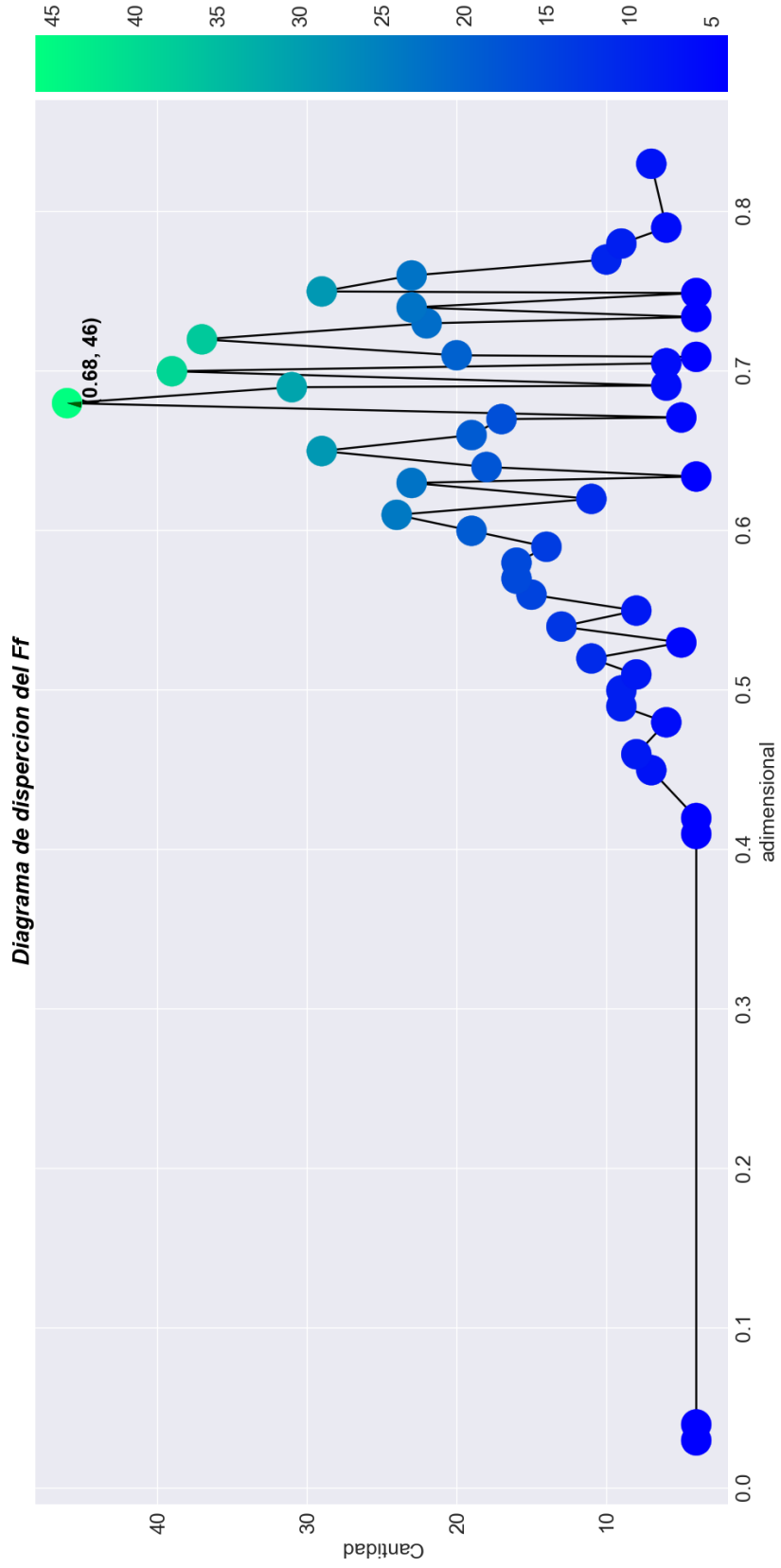

Figura 45. Diagrama de dispersión que permiten observar la frecuencia para los valores del parámetro FF (factor de llenado).