UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Extracción y procesamiento de datos de Amazon para su utilización en un estudio de marketing.

Presentado por Daniel Arnaiz Gutierrez
en Universidad de Burgos — 30 de junio de 2019
Tutores: Dr. José Francisco Díez Pastor
Dr. César Ignacio García Osorio

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación de un proyecto es quizás la fase mas importante de éste. Una buena planificación consigue que el desarrollo del proyecto siga su curso con normalidad y con el mínimo número de imprevistos. A la hora de planificar el proyecto, son varias las partes a tener en cuenta: tiempo, trabajo, y dinero.

A continuación se tratarán los detalles del plan de proyecto llevado a cabo en dos secciones:

Planificación temporal: En este apartado se analizará y explicará cómo se ha planificado el tiempo durante el desarrollo del proyecto teniendo en cuenta el trabajo necesario para cada parte.

Estudio de viabilidad: En este apartado se estudiará la viabilidad del proyecto en distintos campos:

- **Viabilidad económica:** Estimación de los costes y beneficios del proyecto.
- **Viabilidad legal:** Análisis sobre los conceptos legales del proyecto, esto engloba las licencias y políticas utilizadas.

A.2. Planificación temporal

Antes de empezar con el análisis sobre la planificación temporal es necesario añadir que se ha seguido una metodología ágil para la realización del proyecto, más en concreto se ha seguido el método *Scrum*. Es por esto que el desarrollo del proyecto se ha dividido en diferentes *sprints*, con reuniones

entre todos los integrantes del proyecto para tratar los cambios y las siguientes tareas a realizar.

A continuación se mostrarán las principales características de cada *sprint* junto con un gráfico *burndown* generado con la ayuda de la extensión ZenHub y GitHub.

Sprint 1

El primer *sprint* ha resultado ser el más largo. Esto se debe a que el desarrollo del proyecto comenzó mas tarde de lo que en un principio se había planeado. Como se puede ver en el gráfico *burndown* a continuación, en el último tramo del *sprint* es cuando de verdad las tareas propuestas comienzan a darse por completadas.

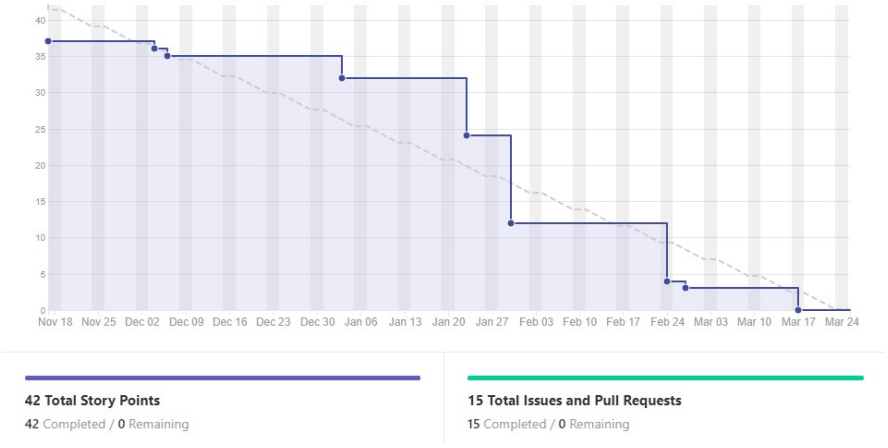

Figura A.1: Gráfico *burndown* del *sprint* 1

Este primer *sprint* se centra principalmente en la puesta en marcha de las principales herramientas a utilizar, además de la toma de algunas decisiones como pueden ser la elección de qué *web scraper* y etiquetador de imágenes utilizar.

Debido a la extensa duración de este *sprint*, al finalizar ya encontramos algunas tareas más avanzadas completadas:

- Primeras versiones de *spiders* funcionales.
- Etiquetado manual de un número reducido de imágenes.
- Extracción de los primeros productos con sus respectivos campos.

Sprint 2

Este *sprint* cuenta con una duración mucho menor al anterior. Es aquí donde comienza la familiarización con LaTeX y las primeras mejoras al *web scraper*.

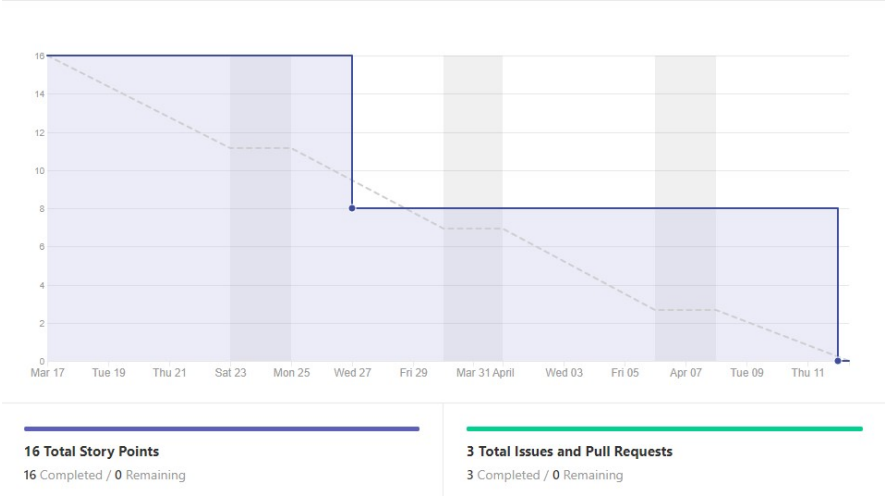

Figura A.2: Gráfico *burndown* del *sprint* 2

Las tareas realizadas en este *sprint* han sido las siguientes:

- Prueba con nuevas etiquetas en el clasificador manual de imágenes.
- Comenzar con la documentación del proyecto.
- Modificación de los campos a extraer en el *web scraper*.

Sprint 3

Este *sprint* se centra en añadir nuevas funcionalidades al proyecto, como pueden ser el almacenamiento en la base datos o la extracción de comentarios.

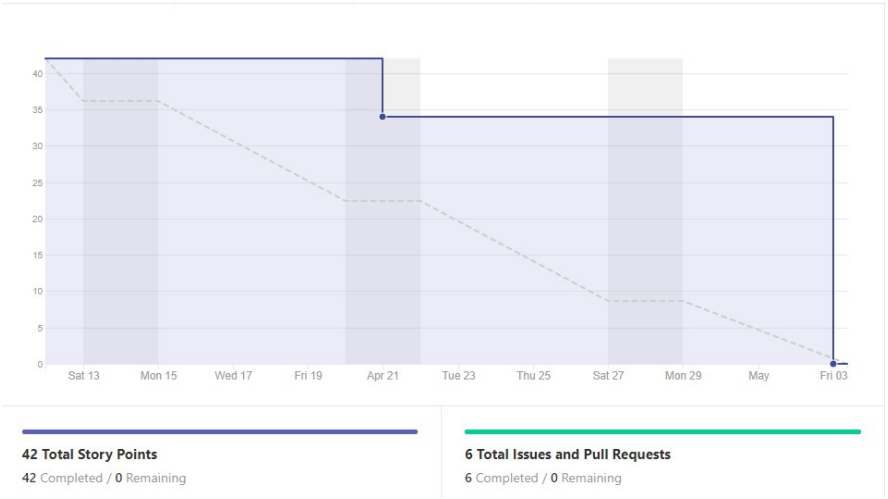

Figura A.3: Gráfico *burndown* del *sprint* 3

Las tareas realizadas en este *sprint* han sido las siguientes:

- Extracción de comentarios asociados a cada producto.
- Comenzar a hacer pruebas con el almacenamiento en una base de datos.
- Estudiar como generar un documento Excel a partir de los datos extraídos.
- Buscar información sobre como aplicar regresión lineal sobre la información extraída.

Sprint 4

Este *sprint* ha estado orientado sobre todo a la investigación de posibles modificaciones y mejoras.

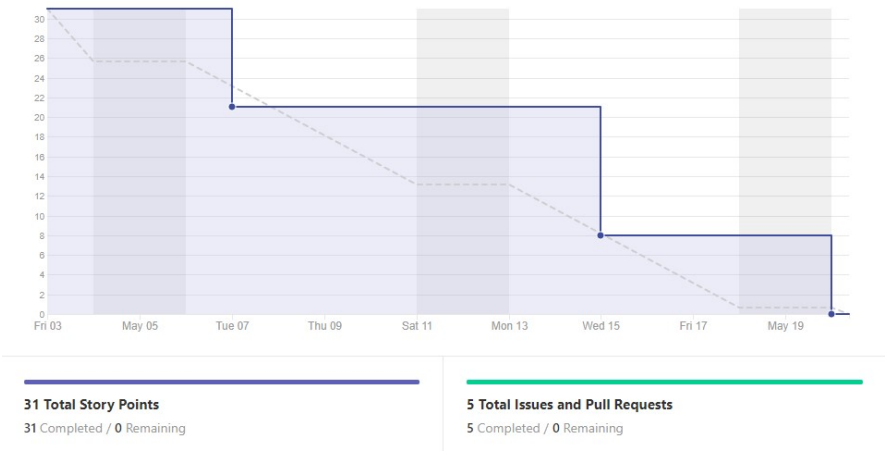

Figura A.4: Gráfico *burndown* del *sprint* 4

Las tareas realizadas en este *sprint* han sido las siguientes:

- Investigación sobre OpenCV y sus librerías de detección facial.
- Primeras iteraciones del clasificador automático de imágenes.
- Probar el funcionamiento del detector facial sobre nuestro propio *dataset*.
- Añadir los enlaces de productos para mujeres en el *web scraper*.
- Añadir un nuevo campo con el sexo al que va dirigido un artículo a la hora de extraer los productos.

Sprint 5

Sprint algo más extenso que se centra principalmente en la primera versión funcional del clasificador automático y la resolución de algunos problemas encontrados.

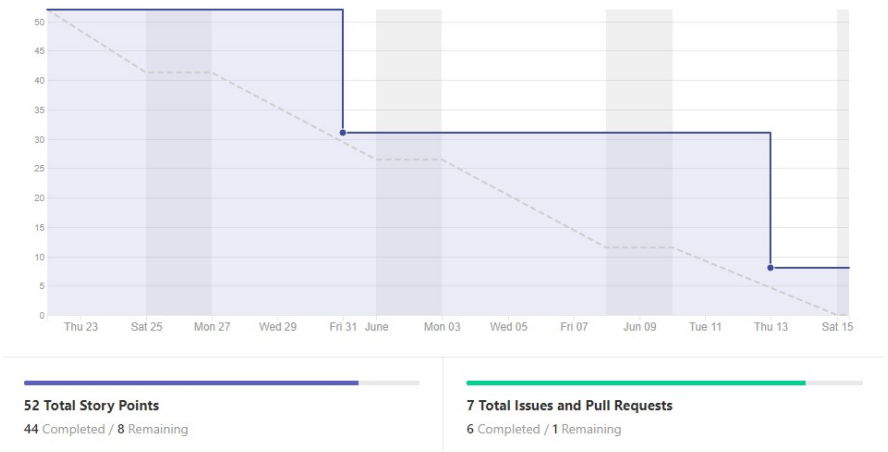

Figura A.5: Gráfico *burndown* del *sprint* 5

Las tareas realizadas en este *sprint* han sido las siguientes:

- Extracción de 1000 productos para la creación del *dataset*a final.
- Primera versión funcional del clasificador automático Modelo/No modelo.
- Primeras pruebas con la detección de caras parcialmente visibles.
- Investigar sobre como generar un domuento Excel a partir de la base de datos.

Sprint 6

Sprint final del proyecto. Es aquí donde se finaliza el proyecto y se hace casi la totalidad de la documentación.

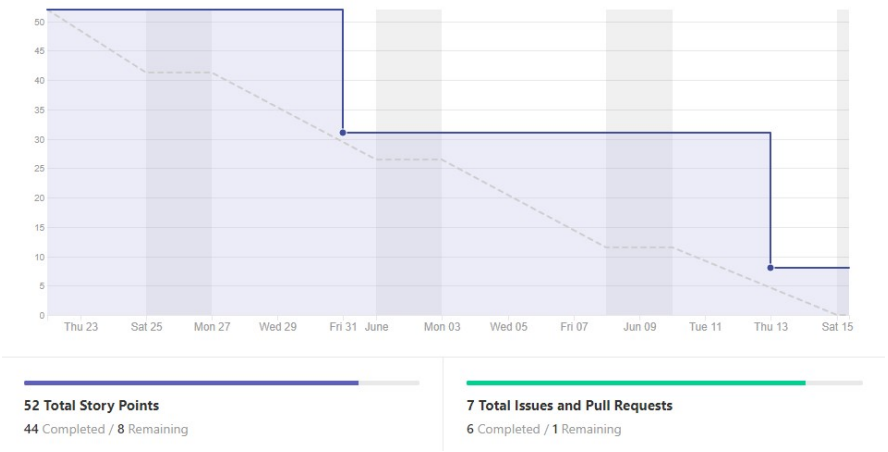

Figura A.6: Gráfico *burndown* del *sprint* 6

Las tareas realizadas en este *sprint* han sido las siguientes:

- Separación del campo que recoge el rango de precios de un producto en dos: precio mínimo y precio máximo.
- Revisión y actualización de los nombres de las tablas y los campos de la base de datos.
- Finalización de la documentación del proyecto y sus respectivos anexos.
- Finalización de los *notebooks* de entrenamiento y clasificación de imágenes.

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se analizará los costes y beneficios estimados de este proyecto en un entorno empresarial.

Costes

Costes de personal

La totalidad de este proyecto ha sido desarrollada por una única persona a tiempo parcial durante un periodo aproximado de 5 meses. Aplicando el

salario mínimo y las cuotas de la Seguridad social[1], se puede estimar un coste de personal de la siguiente forma:

Concepto	Coste (€)
Salario neto	1000 €
Retención IRPF (19 %)	360,53 €
Seguridad social (28,30 %)	537,00 €
Salario total (mensual)	1897,53 €
Total 5 meses	9.487,65 €

Tabla A.1: Costes de personal

Costes de material

En cuanto a los costes materiales y a nivel de programas informáticos, el único gasto es el del ordenador utilizado para la realización del proyecto. Todas las librerías utilizadas son gratuitas y no es necesario la contratación de un servidor para el funcionamiento del proyecto.

Por lo tanto, asociando un valor aproximado de 1300€ al ordenador y estimando una amortización de 5 años, el coste total de materiales para estos 5 meses es el siguiente:

$$\frac{1.300 \text{ €}}{5 \text{ } años * 12 \text{ } meses} = 21{,}67 \text{ €}$$

Concepto	**Coste**	**Amortización**
Ordenador	1.300 €	21,67 €
Total 5 meses	108,34 €	

Tabla A.2: Costes de hardware

Coste total

A continuación se muestra el coste total del proyecto:

[1] http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537

Concepto	**Coste**
Personal	9.487,65 €
Material	108,34 €
Total	9.595,99 €

Tabla A.3: Costes totales del proyecto

Beneficios

En cuanto a los beneficios del proyecto, no se desarrollado teniendo como meta la generación de ingresos. Se podría comercializar cobrando una tarifa mensual a los clientes u ofreciendo un servicio personalizado a cada cliente que dependería del número de productos a extraer.

Viabilidad legal

Para el estudio de la viabilidad del producto, se van a analizar las librerías usadas en el proyecto, anotando las licencias de las que hacen uso. A continuación se listan:

Dependencia	Versión	Licencia
Scrapy	1.5.1	BSD
Pillow	6.0.0	PIL[2]
requests	2.21.0	MIT
js2xml	0.3.1	MIT
Keras	2.3.1	MIT
Numpy	1.16.4	BSD
OpenCV	4.1.0	BSD
JSON	-	FREE
SQLite	3.28.0	FREE
Pandas	0.24.2	BSD

Tabla A.4: Licencias utilizadas

[2] http://www.pythonware.com/products/pil/license.htm

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este apartado recoge los requisitos y objetivos del proyecto. Se detallarán los objetivos generales y tanto los requisitos funcionales como los no funcionales.

B.2. Objetivos generales

El principal objetivo de este proyecto es la creación de varias herramientas capaces de extraer y procesar información sobre un familia de productos concreta de Amazon, en este caso camisetas, para su posterior uso en estudios de mercado.

A su vez este proceso está divido en diferentes fases que podemos identificar como objetivos secundarios:

- Extraer información de los distintos productos. Esta información incluye campos como el nombre de la marca, el rango de precios, número de valoraciones o imágenes del producto.
- Implementar un clasificador automático de imágenes.
- Almacenar la información descargada y las imágenes clasificadas en múltiples tablas con sus respectivos campos dentro de una base de datos.
- Visualizar la información descargada y procesada en un documento Excel.

B.3. Catalogo de requisitos

Requisitos funcionales

A continuación se muestran los requisitos funcionales que han sido implementados en las herramientas creadas para el proyecto:

- **RF 1 Extraer información de los productos**. La herramienta debe ser capaz de extraer los diferentes campos de los productos deseados.
  - **RF 1.1 Extraer los campos principales**. La aplicación debe ser capaz de extraer los campos principales de cada producto.
  - **RF 1.2 Extraer las imágenes**. La aplicación debe de ser capaz de extraer los enlaces de las imágenes asociadas a cada producto.
  - **RF 1.3 Extraer los comentarios**. La aplicación debe ser capaz de extraer los comentarios que los clientes han dejado a cada artículo.
- **RF 2 Almacenamiento en base de datos**. La herramienta debe ser capaz de almacenar la información extraída en una base de datos.
- **RF 3 Generar archivo JSON**. La aplicación debe ser capaz de generar un archivo JSON que contenga de forma estructurada todos los campos extraídos de cada producto.
  - **RF 3.1 Generar un archivo JSON con los campos principales**. La herramienta debe ser capaz de generar un archivo JSON que contenga los campos principales de cada artículo y los enlaces a sus imágenes.
  - **RF 3.2 Generar un archivo JSON con los comentarios**. La herramienta debe ser capaz de generar un archivo JSON que contenga los comentarios que los clientes han dejado a cada artículo.
- **RF 4 Entrenar un clasificador de imágenes**. La herramienta debe poder entrenar un clasificador de imágenes.
  - **RF 4.1 Conjunto de entrenamiento**. La herramienta ha de ser capaz de crear un conjunto de datos de entrenamiento para entrenar los clasificadores.
  - **RF 4.2 Entrenar clasificador modelos**. La herramienta debe de ser capaz de entrenar un clasificador de imágenes que detecte si en una imagen aparece un modelo o no.
  - **RF 4.3 Entrenar clasificador caras**. La herramienta debe ser capaz de entrenar un clasificador de imágenes que detecte si en una imagen la cara de un modelo es visible o no.

- **RF 5 Clasificar automáticamente imágenes**. La herramienta ha de ser capaz de clasificar imágenes de forma automática.
  - **RF 5.1 Clasificador modelos**. La herramienta debe ser capaz de clasificar imágenes en función de si en una imagen aparece un modelo o no.
  - **RF 5.2 Clasificador caras**. La herramienta debe ser capaz de clasificar imágenes en función de si en una imagen la cara de un modelo es visible o no.
- **RF 6 Generar Excel**. La herramienta ha de ser capaz de generar un documento Excel que contenga toda la información sobre los productos extraídos, a partir de la base de datos existente.

Requisitos no funcionales

- **RNF 1 Proceso automático**. Todo el proceso, desde la extracción de datos a la generación del documento final, ha de ser lo mas automatizado posible.
- **RNF 2 Facilidad instalación**. La herramienta debe ser fácil de instalar y ponerse en marcha.
- **RNF 2 Software libre**. La herramienta ha de utilizar software libre.

B.4. Especificación de requisitos

Caso de uso 1: Extraer información de los productos.		
Descripción	Permite al usuario extraer la información los productos deseados.	
Requisitos	RF 1	
	RF 1.1	
	RF 1.2	
	RF 1.3	
Precondiciones	Es necesaria conexión a Internet y los enlaces de los productos a extraer.	
Secuencia normal	Paso	Acción
	1	El usuario indica el fichero que contiene los enlaces de los productos.
	2	Se inicia el proceso de *web scraping*.
	3	Se extrae de forma secuencial todos los campos de todos los productos seleccionados.
	4	Se guarda en la base de datos la información extraída.
Postcondiciones	Se ha extraído y almacenado la información de los productos deseados.	
Excepciones	No se ha podido acceder a «Amazon.com».	
Importancia	Alta	
Urgencia	Alta	

Tabla B.1: Caso de uso 1: Extraer información de los productos.

Caso de uso 2: Almacenamiento en base de datos.		
Descripción	Permite al usuario almacenar la información extraída en una base de datos.	
Requisitos	RF 1	
	RF 2	
Precondiciones	Se debe haber ejecutado el *web scraper*.	
Secuencia normal	Paso	Acción
	1	Se crea una conexión con la base de datos.
	2	Se crean las diferentes tablas y sus campos.
	3	Se almacenan los campos de los productos descargados.
	4	Se cierra la conexión con la base de datos
Postcondiciones	Todos los campos quedan almacenados correctamente.	
Excepciones	Excepción SQL o sintaxis incorrecto.	
Importancia	Alta	
Urgencia	Media	

Tabla B.2: Caso de uso 2: Almacenamiento en base de datos.

Caso de uso 3: Generar archivo JSON.		
Descripción	Permite al usuario extraer la información los productos deseados.	
Requisitos	RF 1	
	RF 2	
Precondiciones	Es necesario disponer de una terminal y conocer la ruta donde se desea guardar el archivo JSON.	
Secuencia normal	Paso	Acción
	1	El usuario indica la ruta del archivo que desea crear.
	2	Se inicia el proceso de *web scraping*.
Postcondiciones	Se genera correctamente el fichero JSON que contiene la información descargada.	
Excepciones	Error de sintaxis.	
Importancia	Media	
Urgencia	Media	

Tabla B.3: Caso de uso 3: Generar archivo JSON.

Caso de uso 4: Entrenar un clasificador de imágenes.		
Descripción	Permite al usuario extraer la información los productos deseados.	
Requisitos	RF 1	
	RF 1.1	
	RF 1.2	
	RF 1.3	
Precondiciones	Es necesario disponer del conjunto de datos de entrenamiento.	
Secuencia normal	Paso	Acción
	1	Se descargan las imágenes etiquetadas manualmente.
	2	Se divide el conjunto en 2 subconjuntos: entrenamiento y test.
	3	Se comienza a entrenar el clasificador a partir de estos conjuntos.
	4	Se descarga el clasificador ya entrenado para su posterior uso.
Postcondiciones	Se entrenado correctamente el clasificador y es capaz de etiquetar las imágenes con un a precisión aceptable.	
Excepciones	Error de sintaxis.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.4: Caso de uso 4: Entrenar un clasificador de imágenes.

Caso de uso 5: Clasificar automáticamente las imágenes.		
Descripción	Permite al usuario extraer la información los productos deseados.	
Requisitos	RF 1	
	RF 1.1	
	RF 1.2	
	RF 1.3	
	RF 3.1	
	RF 4	
	RF 4.1	
	RF 4.2	
	RF 4.3	
Precondiciones	Es necesario disponer de un clasificador previamente entrenado.	
Secuencia normal	Paso	Acción
	1	El usuario indica la ruta de los productos de los que desea clasificar sus imágenes.
	2	El clasificador itera sobre estas imágenes anotando una etiqueta en cada una en función de su clasificación.
	4	Se guardan en la base de datos las anotaciones generadas asociadas a cada producto.
Postcondiciones	Las imágenes han sido clasificadas y las anotaciones almacenadas.	
Excepciones	Error de sintaxis. Error al cargar una imagen. Error al clasificar. Excepción SQL	
Importancia	Alta	
Urgencia	Alta	

Tabla B.5: Caso de uso 5: Clasificar automáticamente las imágenes.

Caso de uso 6: Generar documento Excel.		
Descripción	Permite al usuario generar un documento Excel con toda la información descargada y procesada.	
Requisitos	RF 1	
	RF 1.1	
	RF 1.2	
	RF 1.3	
	RF 3.1	
	RF 4	
	RF 4.1	
	RF 4.2	
	RF 4.3	
Precondiciones	Es necesario disponer de la base de datos completa junto con las predicciones del clasificador de imágenes.	
Secuencia normal	Paso	Acción
	1	Creación de *dataframes* con la ayuda de la librería Pandas a partir de la base de datos.
	2	Generación del documento Excel a partir de los *dataframes* creados anteriormente.
	4	Descarga del documento Excel que contiene toda la información.
Postcondiciones	Se ha generado un documento excel con la información de cada producto y la predicción que ha devuelto el clasificador de imágenes.	
Excepciones	Error de sintaxis. Error al conectar con la base de datos.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.6: Caso de uso 6: Generar documento Excel.

Apéndice C

Especificación de diseño

C.1. Introducción

Este apartado se encarga de recoger los diferentes diseños que han sido llevados a cabo para la realización del proyecto y cumplir de forma satisfactoria los requisitos y objetivos anteriormente tratados.

C.2. Diseño de datos

Esta sección detalla cómo se han almacenado los datos, tanto en la base de datos SQLite, como en los archivos JSON que genera Scrapy.

Estructura JSON

Cada vez que *scrapeamos* un producto, éste será añadido a un archivo JSON que genera Scrapy. En total, se generan 3 archivos JSON por cada uso de la herramienta:

- **Enlaces a cada producto:** El primer JSON recoge todos los enlaces que apuntan a los productos de los cuales se desea extraer la información. La estructura es muy sencilla ya que únicamente guarda un enlace por cada producto:

Figura C.1: Estructura del JSON con los enlaces a los productos

Como se puede ver, cada página de «Amazon.com» contiene 48 productos. En este JSON se almacenan los enlaces a estos 48 artículos.

- **Campos principales de los productos:** Este JSON almacena los campos que se extraen de cada producto de la siguiente forma:

Figura C.2: Estructura del JSON con los campos de cada producto

Como se puede apreciar, la estructura se compone de los siguientes campos:

ASIN: Código identificativo de cada producto.

Sexo: Sexo al que va dirigido cada artículo.

Rango de precios: El precio de cada producto puede variar en función del color o la talla. Este campo recoge el precio máximo y mínimo de cada producto.

Puntuación: Media de las puntuaciones que los clientes han votado a cada producto. Va del 0 al 5.

Número de valoraciones: Número de clientes que han proporcionado una valoración del producto.

Marca: El nombre de la marca de cada artículo.

Descripción: Texto que el vendedor ha utilizado para describir y dar detalles de cada producto.

Enlaces a las imágenes: Enlaces de las imágenes asociadas a cada artículo.

- **Comentarios de los productos:** Por último, este JSON se encarga de recoger los diferentes comentarios que los clientes han dejado a cada artículo. Se compone de dos únicos campos: una lista con los comentarios y el identificador del producto:

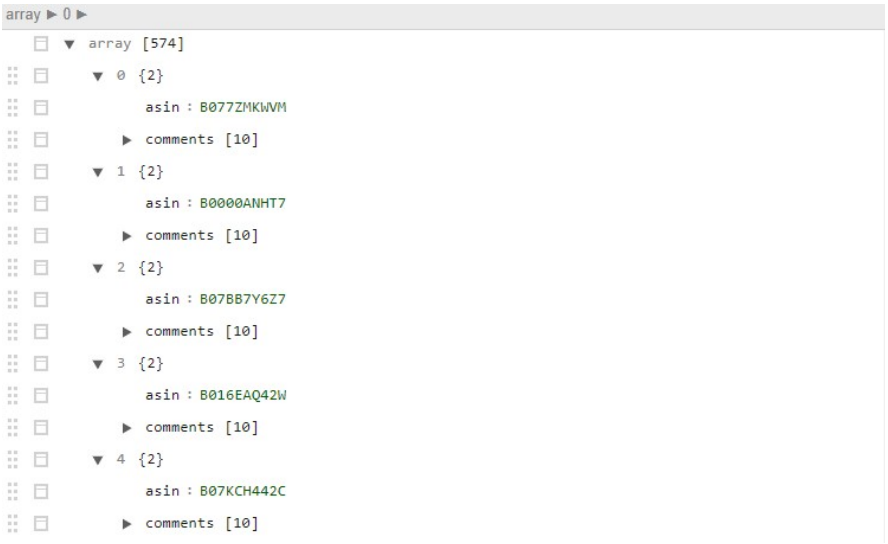

Figura C.3: Estructura del JSON con los comentarios de cada producto

Estructura de la base de datos SQLite

Este proyecto también hace uso de una base de datos para almacenar toda la información una vez han sido procesadas y clasificadas las imágenes. Para ello se hace uso de una base de datos cuyas tablas y campos son los siguientes:

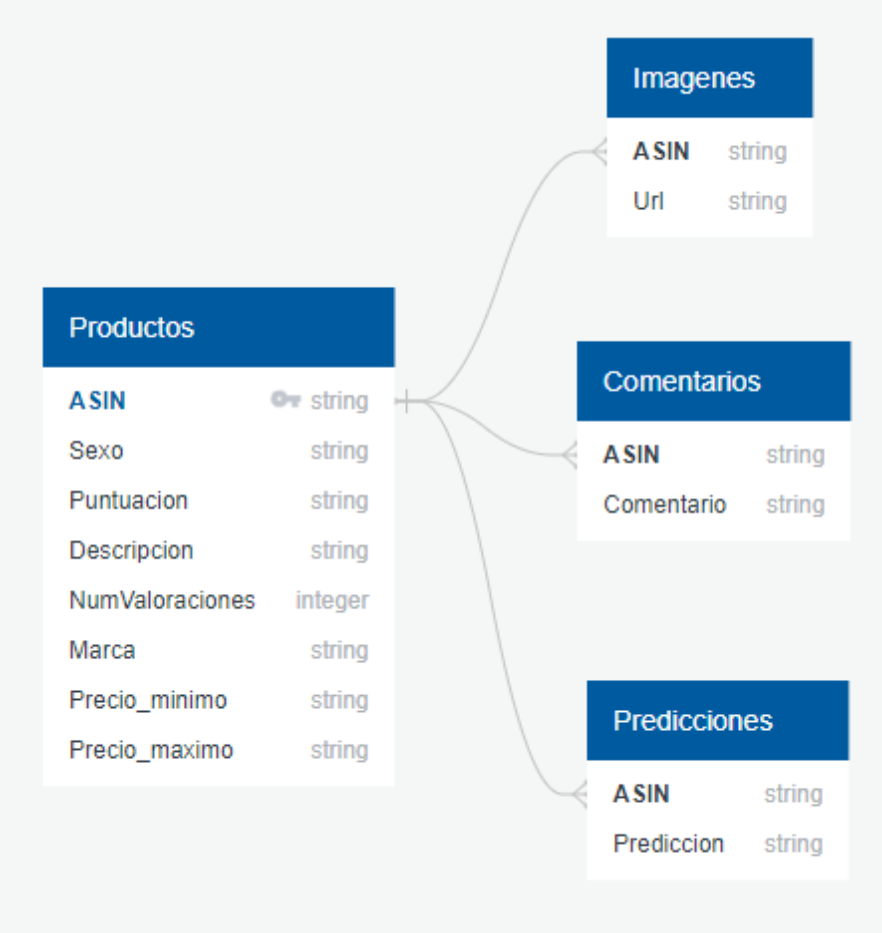

Figura C.4: Diagrama de la base de datos utilizada.

Tabla productos Recoge los principales campos de cada producto. Cada entrada pertenece a un artículo diferente.

Tabla Imágenes Almacena los enlaces a las imágenes de cada producto. Puede haber varias imágenes para un único producto.

Tabla Comentarios Almacena los comentarios asociados a cada artículo. Puede haber varios comentarios de cada producto.

Tabla Predicciones Almacena las predicciones que los clasificadores han generado para la imagen principal de cada artículo. Cada entrada es de un producto diferente.

C.3. Diseño procedimental

En esta sección se muestran los diferentes procedimientos para la realización de los procesos mas importantes del proyecto:

Web scraping

A la hora de la extracción de información de los productos, el proceso a seguir es el siguiente:

1. Extraer los enlaces a dichos productos. Para ello se hace uso del *spider* llamado «urls_spider» en Scrapy.
2. Con el archivo JSON que se ha generado en el anterior paso, se pasará a la extracción de información de cada producto por medio del *spider* llamado «products_spider».
3. Por último, si se desean extraer los comentarios de los productos, se hará uso del *spider* llamado «comments_spider» para ello.

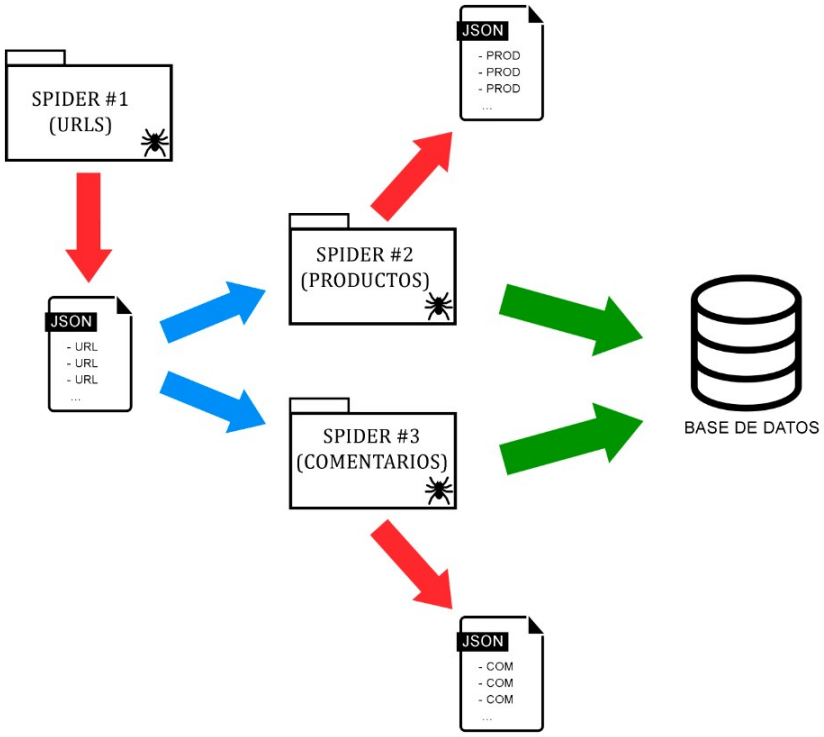

Figura C.5: Proceso de *web scraping* del proyecto

Entrenamiento de los clasificadores

Para entrenar los clasificadores es necesario haber creado un conjunto de entrenamiento y haber etiquetado de forma manual las imágenes que conforman dicho conjunto. Una vez se dispone de ese conjunto de entrenamiento los pasos para entrenar el clasificador son los siguientes:

1. Descargar el *dataset* que contiene las imágenes previamente etiquetadas.
2. De forma secuencial, se convertirá cada imagen a formato *array* y se almacenará junto a su etiqueta.
3. Crear conjuntos de test y entrenamiento para aplicar la técnica de validación cruzada.
4. Inicializar el modelo del clasificador.
5. Entrenar el clasificador.
6. Almacenar el clasificador ya entrenado para su posterior uso.

Clasificación de imágenes y almacenamiento de resultados

El proceso de clasificación de imágenes y su posterior almacenamiento es el siguiente:

1. Importar y cargar los clasificadores previamente entrenados.
2. Importar el archivo JSON que contiene los productos a clasificar.
3. Descargar cada imagen con el ASIN del producto como nombre.
4. Clasificar cada imagen siguiendo el siguiente algoritmo:

```
if Imagen cargada correctamente then
    Redimensionar imagen.
    Convertir imagen a formato array.
    Aplicar clasificador que detecta modelo.
    if El clasificador ha encontrado un modelo en la imagen then
        Aplicar clasificador que detecta cara.
        if El clasificador ha encontrado la cara del modelo then
            Devolver predicción: Modelo con cara.
        end
        Devolver predicción: Modelo sin cara.
    end
    Devolver predicción: Sin modelo.
end
```

Algoritmo 1: Algoritmo de clasificación de imágenes

5. Importar la base de datos
6. Crear la tabla «Predicciones» en la base de datos
7. Almacenar en la base de datos cada predicción generada por el clasificador quedando asociada al artículo al que pertenece la imagen.

C.4. Diseño arquitectónico

En cuanto al diseño arquitectónico del proyecto, es necesario señalar que la primera parte del proyecto, todo lo relacionado con la extracción de datos y *web scraping*, funciona bajo la estructura se Scrapy. El resto se encuentra dividido en dos *notebooks* de Google Colab, el primero para el entrenamiento de los clasificadores, y el segundo destinado a la clasificación de imágenes y su posterior almacenamiento en la base de datos.

El siguiente diagrama muestra una visión general de la arquitectura de Scrapy junto con los conponentes que lo conforman y el flujo de datos que se lleva a cabo en el sistema [1]:

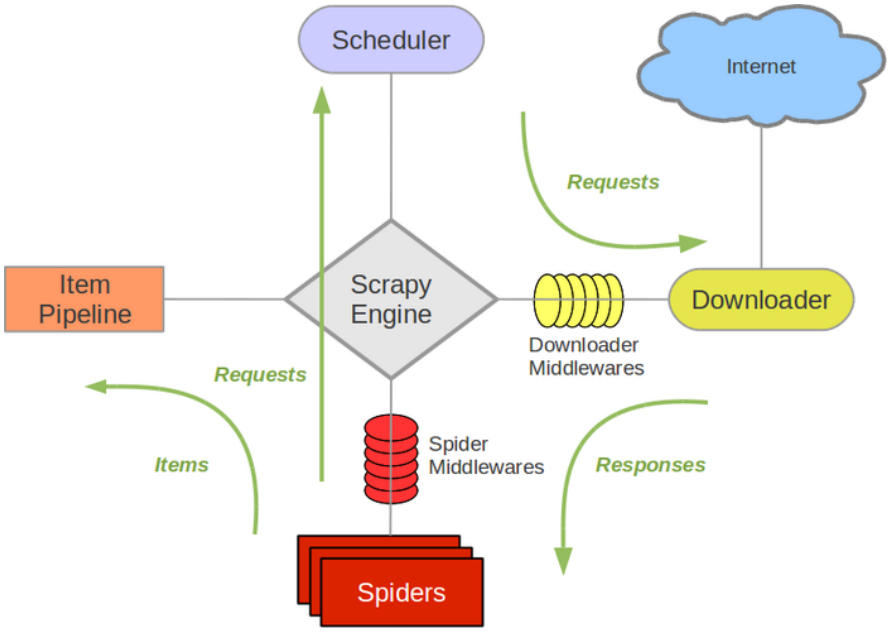

Figura C.6: Arquitectura que sigue Scrapy y sus diferentes componentes

Componentes de scrapy

A continuación se describen los principales componentes de la arquitectura de un proyecto de Scrapy:

- ***Items*:** Los diferentes elementos o campos a extraer.
- ***Spiders*:** Clases escritas por el usuario que contienen los diferentes procedimientos para parsear y extraer los *items* de uno o varios enlaces. En este caso hemos hecho uso de 3 *spiders*: Una para obtener los enlaces a cada artículo, otra para extraer los comentarios, y la última para el resto de campos.
- ***Pipelines*:** Apartado responsable de procesar los *items* una vez han sido extraidos por las *spiders*. En este proyecto se han usado para almacenar los campos extraídos en una base de datos.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Esta sección se encarga de detallar los conceptos relacionados con la documentación técnica que sigue el proyecto. Se tratará de explicar la estructura de directorios que sigue el proyecto, así como los pasos a seguir para ser instalado y ejecutado.

D.2. Estructura de directorios

A continuación se muestra la estructura de directorios en la que se distribuye el proyecto:

- **Scrapy/** Directorio principal de Scrapy. Se compone de los siguientes archivos y subdirectorios:
  - **amazon/spiders/** Directorio donde se encuentran las diferentes *spiders*.
    - **urls_spiders.py** *Spider* utilizada para extraer los enlaces de los productos.
    - **products_spiders.py** *Spider* utilizada para extraer los principales campos de cada producto.
    - **comments_spiders.py** *Spider* utilizada para extraer los comentarios asociados a cada producto.
  - **amazon/items.py** Fichero que recoge los diferentes campos que extrae Scrapy.

  - **amazon/middlewares.py** Fichero que recoge la configuración de los *middlewares* se Scrapy si los hubiera.
  - **amazon/pipelines.py** Fichero que se encarga del almacenamiento en la base de datos de los campos extraídos.
  - **amazon/settings.py** Fichero de configuración del proyecto de Scrapy.

- **Documentacion/** Este directorio contiene todos los archivos relacionados con la documentación y la memoria del proyecto, Tanto los ficheros fuente de LaTeX, como los PDFs ya finalizados.

- **Colab notebooks/** Directorio donde se encuentran los *notebooks* de Google Colab utilizados en el proyecto:

  - **Clasificador_Modelo_Sin_modelo.ipynb** *Notebook* encargado del entrenamiento del clasificador de imágenes que detecta si hay un modelo en la imágen o no.
  - **Clasificador_cara_sin_cara.ipynb** *Notebook* encargado del entrenamiento del clasificador de imágenes que detecta si la cara de un modelo es visible o no.
  - **Clasificador_final_excel_db.ipynb** *Notebook* encargado de la clasificación de imágenes y su posterior almacenamiento en la base de datos y documento Excel.

- **Databases/** Carpeta que contiene las diferentes bases de datos utilizadas durante la realización del proyecto.

- **Datasets/** Directorio que recoge los *datasets* utilizados para el entrenamiento de los clasificadores:

  - **dataset-modelo.zip** *Dataset* que contiene las imágenes ya etiquetadas utilizadas para el entrenamiento del clasificador de imágenes que detecta modelos.
  - **dataset-cara.zip** *Dataset* que contiene las imágenes ya etiquetadas utilizadas para el entrenamiento del clasificador de imágenes que detecta caras.

- **Keras models/** Directorio donde encontramos los clasificadores de Keras ya entrenados para su posterior utilización:

  - **clasificador-modelo.zip** Clasificador de imágenes encargado de detectar modelos ya entrenado y listo para su utilización.
  - **clasificador-cara.zip** Clasificador de imágenes encargado de detectar caras ya entrenado y listo para su utilización.

- **Outputs/** Directorio donde se guardan todos los resultados del proyecto: Archivos JSON y documentos Excel.
- **Scripts/** Directorio que incluye algunos *scripts* utilizados para algunas tareas del proyecto:
  - **JSONmerger.py** *Script* que combina los campos de dos archivos JSON utilizando el campo «url» como elemento identificador.
  - **dataextractor.py** *Script* que descarga imágenes dada su URL y las renombra en funcion de cómo han sido etiquetadas. Utilizado para la creación de los *datasets*.
  - **dataturks_fixer.py** *Script* usado para arreglar el JSON generado por Dataturks para su correcto funcionamiento.
  - **urlextractor.py** *Script* que extrae los enlaces de las imágenes de cada producto y lo añade a una lista para posteriormente etiquetarlo en Dataturks.

D.3. Manual del programador

Apartado que detalla algunos de los procesos llevados a cabo en el proyecto necesarios para que pueda seguir desarrollándose.

Manual de Scrapy

Para la creación del proyecto de Scrapy es necesaria la instalación de la librería. Esto se puede hacer de varias formas, tal y como describe la documentación oficial[1]:

- **Utilizando Conda:** Se instala la librería con el siguiente comando:

```
conda install -c conda-forge scrapy
```

- **Utilizando PIP:** Se instala la librería con el siguiente comando:

```
pip install Scrapy
```

Una vez instalado el paquete de Scrapy, la creación de un nuevo proyecto es sencilla:

[1] http://doc.scrapy.org/en/latest/intro/install.html

```
scrapy startproject [nombre_proyecto]
```

Este comando se encarga de crear la estructura de carpetas necesaria para el correcto funcionamiento de Scrapy. A continuación se detalla el proceso a seguir para añadir *spiders* al proyecto:

```
cd [nombre_proyecto]
scrapy genspider [nombre_spider] [direccion_web]
```

En cuanto a la ejecución de un *spider*, se hace con el siguiente comando:

```
scrapy crawl [nombre_spider] -o [nombre_output]
```

Manual de Dataturks

A continuación se detallará el proceso a seguir para el etiquetado manual de imágenes con la ayuda de la herramienta Dataturks[2].

1. Iniciar sesión por medio de GiHub o creando una cuenta a parte.
2. Crear un nuevo *dataset* y elegir el tipo adecuado, en este caso será el llamado *Image Classification*.

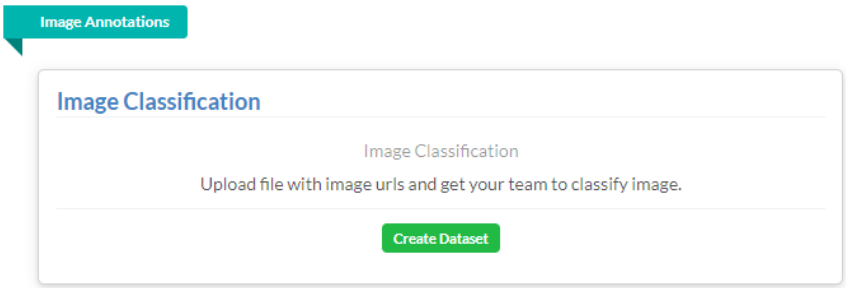

Figura D.1: Tipo de *dataset* a crear en Dataturks

3. Configurar el funcionamiento del *dataset*

[2] https://dataturks.com/

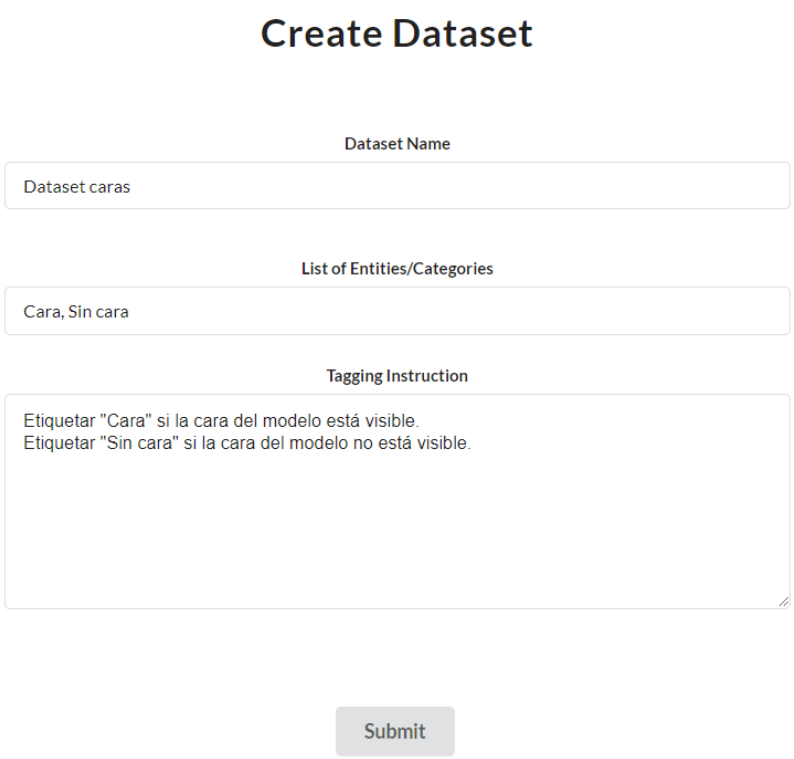

Figura D.2: Creación de un *dataset* en Dataturks

4. Importar una lista con los enlaces a las imágenes que se desean etiquetar. Deberán estar en un documento *txt*.

5. Una vez subidas las imágenes, se puede empezar a etiquetar seleccionando la etiqueta adecuada en cada caso.

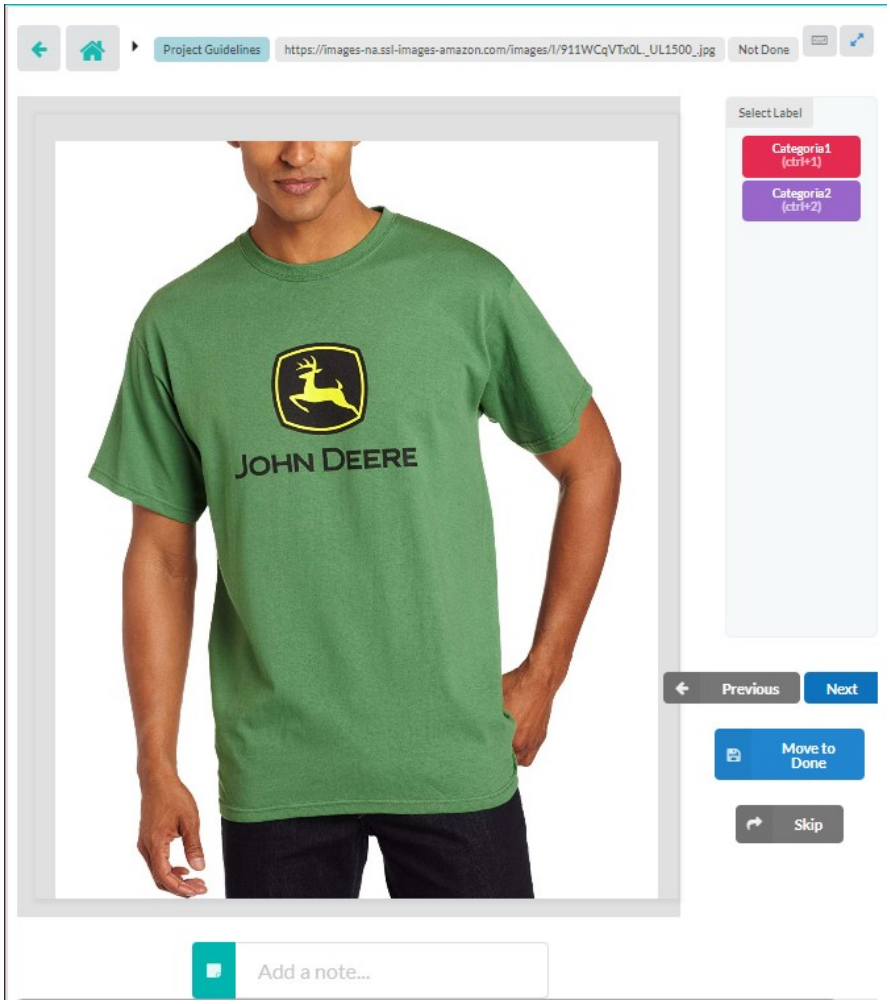

Figura D.3: Proceso de etiquetado manual en Dataturks

6. Por último, para exportar los resultados simplemente se ha de descargar el archivo generado pulsando en el botón *Download*.

Entrenamiento de los clasificadores

El proceso de entrenamiento de los clasificadores es muy sencillo ya que está muy automatizado. Los *notebooks* se encargan de descargar y descomprimir el *dataset* previamente subido al repositorio del proyecto para así entrenar el clasificador. Una vez entrenado, se muestra la precisión alcanzada y un resumen de las capas utilizadas por el modelo de Keras.

Por último se descarga el clasificador ya entrenado para su posterior uso.

D.4. Compilación, instalación y ejecución del proyecto

Lo más importante a la hora de instalar y ejecutar este proyecto es la parte de Scrapy. Puesto que en el repositorio ya se encuentra el proyecto de Scrapy al completo, lo primero será descargar el contenido del repositorio. Para ello se utiliza el siguiente comando:

```
git clone https://github.com/daniarnaizg/TFG-Amazon-Scraper.git
```

Con esto, quedará descargado el repositorio. Lo siguiente será instalar el paquete de Scrapy si no se tiene instalado (Ver manual de Scrapy en la sección anterior).

También es necesaria la instalación de dos paquetes, se pueden instalar por medio de PIP:

```
pip install js2xml
pip install pypiwin32
```

A partir de aquí simplemente se puede lanzar una *spider*, tal y cómo se ha mostrado antes. Es necesario indicar el fichero al que se desea exportar el contenido descargado en formato JSON:

```
scrapy crawl urls_spider -o urls_output.json
scrapy crawl products_spider -o products_output.json
scrapy crawl comments_spider -o comments_output.json
```

En cuanto a la ejecución de los *notebooks*, esto se hará importandolos en el entondo de Google Colab a través del siguiente enlace:

https://colab.research.google.com/

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Scrapy Docs. Scrapy - architecture overview, 2018. https://docs.scrapy.org/en/latest/topics/architecture.html.