INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA

ESCUELA DE POSTGRADO

MODELO DE CATEGORIZACIÓN INTELIGENTE DE PRODUCTOS PUBLICADOS EN LA PLATAFORMA DE UN E-COMMERCE

AUTOR: González Rodríguez Rubén Darío (Leg. Nº 104628)

TUTOR: Pampliega Juan Martín

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE ESPECIALISTA EN CIENCIA DE DATOS

BUENOS AIRES

SEGUNDO CUATRIMESTRE, 2022

Tabla de contenidos

1. Introducción
2. Marco Conceptual
3. Antecedentes
4. Definición del Problema
5. Justificación del estudio
6. Alcances del trabajo y limitaciones
7. Hipótesis
8. Objetivos
9. Tecnologías y herramientas
10. Metodología de solución propuesta
11. Implementación de la solución
12. Resultados y análisis
13. Conclusiones
14. Referencias-Bibliografía

1. Introducción

Una de las actividades que más se ha acentuado en la cotidianidad de la era digital actual es la realización de compras por internet. Cada vez más surgen nuevas aplicaciones y servicios orientados a brindar esta facilidad de comprar productos sin la necesidad de ir de forma presencial al local físico. Este fenómeno ha abierto una ventana de oportunidades para la creación de empresas cuyos modelos de negocio giran en torno a la intermediación entre el cliente y el producto mediante plataformas digitales de comercio en línea, que son los denominados comercios electrónicos (e-commerces).

Debido al elevado número de este tipo de empresas que han surgido dentro del medio, se ha generado un ambiente de alta competitividad en donde el factor diferenciador es la facilidad con la que el usuario puede cumplir su necesidad mientras que tiene la mejor experiencia posible. Para lograr esto, las diferentes plataformas explotan los datos que los mismos usuarios generan cuando las usan, de manera que puedan brindar un mejor servicio mediante sugerencias anticipadas que son producto del análisis de patrones de comportamiento del usuario o de otros muchos con características similares. Un ejemplo de esto es cuando al comprar un paquete de pañales para bebé en una página digital, antes de proceder a pagar es la propia página la que sugiere comprar también un paquete de toallas húmedas; esto debido a que otros usuarios que también compraron pañales los acompañaron con toallas húmedas, lo que se convierte en un plus en caso de que el usuario los estuviese olvidando.

Existen distintas finalidades para las que estas plataformas explotan los datos de sus usuarios, sin embargo, la mayoría están orientadas hacia la mejora del servicio que prestan en la medida en que el usuario perciba mayor facilidad y comodidad para satisfacer su necesidad al usarlas. Dentro de las características que se ofrecen generalmente para mejorar la experiencia del usuario en aplicaciones de compra por internet, y que son producto de la explotación y análisis de datos para identificar patrones, se encuentran:

- Listado de coincidencias más relevantes al realizar una búsqueda.
- Clasificación de productos en categorías.
- Sugerencia o recomendación de productos adicionales.
- Estimación de precios de envíos, costos de viajes, etc.

Particularmente, este proyecto se centra en la funcionalidad de **clasificación de productos en categorías** como eje principal de estudio. Esto debido a que es una característica que es común en la mayoría de plataformas de compras online independientemente del rubro al que pertenezcan, y de la cual dependen muchas de las otras funcionalidades que las hacen atractivas para los usuarios. Por ejemplo, en una

aplicación de “delivery” de comidas es importante que si el usuario selecciona la categoría “pizzas” solo se muestren restaurantes que venden este producto, de lo contrario se puede perder el interés en el uso de dicha aplicación por no acertar en los resultados que muestra según la necesidad del cliente.

La tarea de clasificar productos en categorías resulta extremadamente sencilla de realizar o revisar para un operador humano, sin embargo, el desgaste asociado a esta empieza a tomar un peso importante cuando se debe realizar para cientos de miles de productos que se ofrecen, por ejemplo, en una plataforma de compra y venta de productos online. Es aquí donde se presenta la oportunidad de utilizar técnicas de análisis y minería de datos acompañadas de aprendizaje de máquina para relevar al operador humano de la realización continua de esta tarea de bajo nivel, mientras que se realiza de una manera más rápida y eficiente a la vez que se aprovecha el recurso humano para actividades de mayor nivel.

En este proyecto se propone una solución a la necesidad de automatizar la categorización de productos desde el enfoque de la ciencia de datos mediante la utilización de procesamiento de lenguaje natural (NLP) y aprendizaje de máquina. El caso de estudio analizado corresponde a los datos de productos publicados en la plataforma de un sitio de comercio electrónico (e-commerce), para los cuales se conoce el **título** de la publicación y la **categoría** a la que corresponde, lo que define un problema de clasificación supervisado.

El modelo implementado se soporta sobre una arquitectura de “Big Data” mediante la utilización de tecnologías de almacenamiento y procesamiento distribuidas. Lo anterior constituye uno de los criterios de diseño fundamentales del modelo debido que este fue concebido para responder adecuadamente a las exigencias y requerimientos de un entorno real de operación de un e-commerce. En donde el volumen de información manejado es muy grande debido a los cientos de miles de publicaciones nuevas que se crean diariamente en sus plataformas. Por lo que la utilización de un entorno distribuido y fácilmente escalable es la manera adecuada de afrontar el desafío previamente descrito.

2. Marco Conceptual

Esta sección se encuentra divida en tres partes con la finalidad de que los conceptos y definiciones que aquí se presentan se agrupen para dar respuesta a tres grandes interrogantes. El primero es ¿En qué contexto se ubica el proyecto desarrollado?, el cual se plantea con la finalidad de brindarle al lector las bases para conocer en **donde** se ha centrado el desarrollo realizado. El segundo interrogante es ¿Cuál es la problemática dentro del contexto sobre la cual se centra el proyecto?, con el que se pretende definir en **qué** se trabajó. El ultimo interrogante es ¿De qué manera se puede solucionar dicha problemática?, mediante el cual se introducen los conceptos que permiten al lector entender **cómo** se abordó la solución al problema.

2.1. Conceptos asociados al contexto del problema

2.1.1. Plataformas de comercio electrónico (e-commerces)

Según (Kurtz, 2016), una definición bastante acertada de las plataformas de comercio electrónico (e-commerce) sería que son empresas cuyo modelo de negocio radica en el desarrollo, implementación, operación y mantenimiento de un sitio (plataforma) especializado en el fomento de actividades de compra y venta de productos o servicios a través de internet.

Figura 2-1. Ejemplo ilustrado de la compra de productos a través de una plataforma en internet. Fuente: Freepik (REF)

La utilización de estas plataformas simplifica el proceso de adquisición para el comprador dado que este puede realizar la compra desde la comodidad de su hogar, es decir, sin tener que desplazarse hasta un local físico; siendo los únicos requerimientos contar con un dispositivo que pueda conectarse a internet y poseer un medio de pago habilitado para tal fin (ver figura 2-1). Así mismo, vender productos a través de estos sitios también beneficia a los vendedores debido a que no se hace necesario contar con un local físico, lo que permite ahorrar en gastos de impuestos, alquileres, personal, entre otros. Generalmente, las partes involucradas en una compra en línea no llegan ni

siquiera a conocerse, esto debido a que el envío del producto suele realizarse mediante servicios de mensajería.

Así como los mencionados también existen muchos otros beneficios asociados al uso de estas plataformas, los cuales sirven para explicar el hecho de que los e-commerces se hayan convertido en una de las opciones más utilizadas en los últimos tiempos para la compra y venta de productos y servicios.

2.1.2. Publicaciones en los e-commerces

El funcionamiento de todos los sitios de compra y venta en línea es prácticamente el mismo, todo inicia con los vendedores que desean ofrecer sus productos utilizando estas plataformas, para lo cual deben crear una **publicación** en donde se describan los detalles del producto ofrecido y se especifique el precio por el cual desean venderlo. Estas publicaciones constituyen entonces el canal principal por el que se ponen en contacto compradores y vendedores, por lo que representan una de las bases principales sobre las que se soporta la operación de los e-commerces.

Continuando con el proceso, cuando existe un comprador que accede al sitio para satisfacer una necesidad, este realiza la búsqueda de las publicaciones que ofrecen el producto o servicio deseado. En este punto el cliente tiene la opción de elegir, de entre todas las publicaciones que ofrecen lo requerido, aquella con la cual se sienta más cómodo en función del precio, confianza que inspire el vendedor, entre muchas otras razones.

De todo lo anterior se evidencia la importancia que tienen las publicaciones dentro del proceso, razón por la cual estas deben ser lo más detalladas posibles. A continuación, se describen algunos de los elementos básicos que componen una publicación en la plataforma de un e-commerce.

Figura 2-2. Ilustración de la interfaz de publicaciones típica de un e-commerce.
Fuente: Mercado Libre (REF)

En la figura 2-2 se observa que, en general, cuando se listan las publicaciones más relevantes asociadas a un producto buscado estas se componen de las siguientes cuatro (4) partes:

1) **Título:** Presenta el nombre del producto ofrecido.
2) **Imagen:** Muestra la apariencia que tiene el producto y suele realizarse mediante la utilización de fotografías.
3) **Calificación del vendedor:** Puntuación que permite al comprador conocer la evaluación que otros usuarios han dado sobre el vendedor de la publicación.
4) **Detalles del producto:** Se utilizan para describir las principales características del producto ofertado.

En caso de que una publicación logre captar la atención del cliente con la información presentada en las partes descritas anteriormente, este puede ingresar a dicha publicación para obtener información mucho más detallada del producto así como de las condiciones de la venta, los medios de pago, la forma de envío y mucho más; lo que al final será determinante para que el comprador tome la decisión definitiva de realizar la compra o no hacerla.

2.2. Conceptos asociados a la definición del problema

2.2.1. Categorización de productos en plataformas de comercio electrónico (e-commerce)

Si bien se explicará de manera detallada en una sección posterior de este documento, la problemática abordada en este proyecto está relacionada con la tarea de clasificar los productos ofrecidos en las publicaciones dentro de una serie predefinida de categorías. Esta tarea es inicialmente realizada por el vendedor en el momento de la creación de la publicación, sin embargo, debe ser luego revisada y validada por parte de la plataforma para verificar que el producto se encuentre correctamente clasificado.

Lo anterior es de vital importancia para el funcionamiento del sitio ya que muchas de las funciones que en este se ofrecen están relacionadas con la categorización de los productos publicados. La funcionalidad más representativa asociada a la tarea mencionada es la selección y presentación de publicaciones relacionadas o afines a una búsqueda ingresada por el usuario, por ejemplo, si el usuario realiza una búsqueda de productos de jardinería, este no está esperando que la plataforma le presente publicaciones con productos de la categoría de ropa para bebés. Errores como el mencionado anteriormente pueden ocurrir cuando los productos no se encuentran asignados correctamente a la categoría que pertenecen, lo que impacta negativamente en la experiencia del usuario y no favorece al sitio.

La tarea de asignar la categoría a un producto publicado, tal y como se mencionó anteriormente, la realiza inicialmente el vendedor al crear la publicación, sin embargo, **siempre** debe ser validada por parte de la plataforma de e-commerce; lo cual puede realizarse mediante algún recurso humano o de manera automática utilizando motores de reglas e incluso modelos de aprendizaje de máquina.

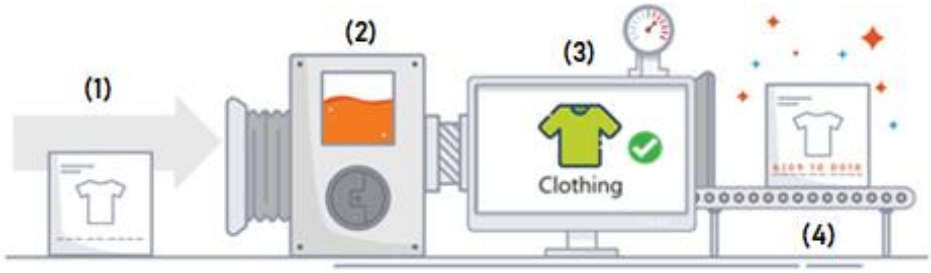

Figura 2-3. Ejemplo ilustrativo del proceso de categorización de los productos publicados. Fuente: Avalara (REF)

La figura 2-3 presenta una ilustración a modo de ejemplo del proceso de categorización de productos en e-commerces. Se empieza con la asignación de la categoría por parte del vendedor en la creación de la publicación (paso 1), seguidamente se realiza la revisión por parte de la plataforma utilizando alguno de los métodos mencionados anteriormente (paso 2), como resultado se realiza la validación o corrección de la categoría del producto (paso 3) y se actualiza en el sitio web de la plataforma (paso 4).

2.3. Conceptos asociados al enfoque utilizado para la solución del problema

2.3.1. Procesamiento de lenguaje natural (NLP por sus siglas en inglés)

Se empieza recordando al lector que, se conoce como "lenguaje natural" al lenguaje que utilizan los seres humanos para satisfacer la necesidad de comunicarse entre sí, el cual es empleado de manera general y rutinaria en el día a día. Este lenguaje puede emplearse en forma oral, escrita o de señas y agrupa a todos los idiomas existentes (español, inglés, portugués, etc).

En (Bird, Steven; Klein, Ewan; Loper, Edward 2009) se define el procesamiento de lenguaje natural como cualquier manipulación computacional realizada sobre datos que representan el lenguaje natural humano, de manera que a la máquina le sea posible "entender" y dar contexto a la información del lenguaje contenida en los datos. Este "entendimiento" no es más que la identificación de patrones presentes en los componentes del lenguaje expresados ya sea de forma escrita o verbal, lo cual se realiza mediante representaciones numéricas de dichos componentes orientadas a capturar las interacciones entre estos que definen una salida particular conocida.

Ejemplos ampliamente conocidos de las operaciones comúnmente realizadas para representar los componentes del lenguaje y facilitar la identificación de patrones contenidos en estos serían la "**tokenización" de palabras**, en donde se asume que cada palabra es un "token" que representa un "feature" para el modelo y se cuentan las repeticiones de cada "token" para intentar capturar la importancia de estos dentro del mensaje transmitido. Y También la **"generación de n-gramas"**, donde se obtienen "features" que son agrupaciones de los diferentes "tokens" de manera que se pueda capturar la interacción de los "tokens" agrupados y el peso de estos dentro de la salida del modelo. Los ejemplos mencionados son solo algunos dentro de las muchas maneras de representar los componentes del lenguaje para que la máquina pueda "entender" la información contenida en los datos y tomar una decisión basada en los patrones identificados.

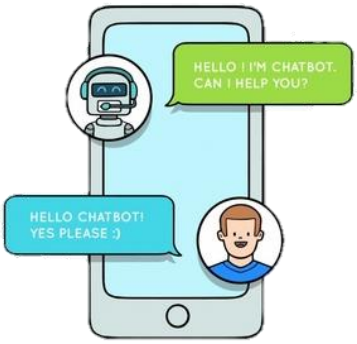

Figura 2-4. Ejemplo ilustrativo de una de las aplicaciones más ampliamente conocidas del NLP. Fuente: Freepik (REF)

En la figura 2-4 se presenta un ejemplo ilustrado de un **chatbot**, una de las aplicaciones más utilizadas y difundidas del NLP. En los chatbots el humano interactúa a través de mensajes de texto con un bot (máquina), el cual debe ser capaz de "entender" e interpretar la información transmitida por el humano en sus mensajes para brindar una respuesta acorde a la necesidad de este. Otro ejemplo conocido son los asistentes virtuales tales como **Siri** de Apple, los cuales utilizan NLP para interpretar y ejecutar ordenes transmitidas por los humanos a través de comandos de voz.

En general, el NLP representa una herramienta para que los modelos de aprendizaje de máquina puedan analizar y reconocer patrones presentes en datos que contienen elementos del lenguaje humano. Por lo que esta herramienta suele utilizarse como parte de la etapa de pre-procesamiento de los datos para generar los "features" que ingresarán al modelo para el proceso de entrenamiento de este.

2.3.2. Aprendizaje de máquina (Machine Learning)

El aprendizaje de máquina es una herramienta perteneciente al área dentro del campo de la **Inteligencia Artificial** conocida como **minería de datos (data mining)**, la cual

tiene como objeto de estudio la extracción de conocimiento e información útil a partir de grandes volúmenes de datos.

De acuerdo con (Grosan & Abraham, 2011), se define como aprendizaje de máquina al conjunto de técnicas, algoritmos y, de manera general, programas de computadora, que son capaces de "aprender" desde la experiencia para mejorar su desempeño al realizar una tarea particular. Este proceso de aprendizaje se lleva cabo de forma iterativa y consiste en presentarle a la máquina un conjunto de casos de ejemplo cuyos resultados son conocidos, de manera que esta sea capaz de identificar los patrones en los datos que determinan los resultados previamente presentados y, poder finalmente realizar predicciones en datos nuevos y con resultados desconocidos.

Desde una perspectiva más técnica, el aprendizaje previamente explicado tiene lugar mediante modificaciones de los parámetros internos de los algoritmos que componen esta disciplina, dichos parámetros se van ajustando durante las sucesivas iteraciones para poder modelar de manera acertada el comportamiento de los datos que están siendo presentados. Finalmente, la convergencia del proceso iterativo arroja un modelo "entrenado" que ha "aprendido" para responder con el mejor desempeño alcanzado a los nuevos casos que deseen evaluarse del problema en cuestión.

2.3.3. Aprendizaje supervisado

Corresponde a un enfoque dentro del aprendizaje de máquina para el que el proceso de aprendizaje se realiza tal cual como fue descrito anteriormente, es decir, mediante la presentación sucesiva de ejemplos cuya respuesta (salida) es conocida (Harrington, 2012). Por lo que la técnica (algoritmo) empleada debe ajustar sus parámetros internos para que al ingresar una muestra con salida conocida, el resultado obtenido corresponda efectivamente con la respuesta esperada. A este proceso se le conoce como "ajustar el comportamiento de los datos", dado que el modelo resultante se fue refinando iterativamente para entregar resultados acordes a los casos conocidos, finalizando el proceso de entrenamiento cuando el desempeño en la salida tiene un valor adecuado. La palabra "supervisado" hace referencia a que para cada ejemplo utilizado en el entrenamiento se conoce la salida que le corresponde, por lo que es posible evaluar el desempeño del modelo obtenido mediante una comparación entre los resultados que entrega y los que se conocen previamente para las muestras con las que fue entrenado.

Es válido mencionar que, si bien el entrenamiento supervisado se realiza con ejemplos cuya salida es conocida, la idea es obtener un modelo que no solo presente buen desempeño para casos conocidos sino que también sea capaz de acertar en la salida cuando se le presentan nuevos casos que nunca antes ha visto.

Figura 2-5. Ejemplo ilustrativo del proceso de aprendizaje supervisado. Fuente: Adobe (REF)

En la figura 2-5 se observa que a la máquina del ejemplo se le presenta un conjunto de datos con objetos para los que conoce su correspondiente etiqueta (manzana, balón y taza), esta máquina toma estos datos y entrena un modelo predictivo para ajustar el comportamiento de estos, lo que quiere decir que se obtiene un modelo que intenta asignar la etiqueta correcta cuando se le muestra un objeto de alguna de las categorías aprendidas. Con el modelo entrenado, se procede a evaluar el desempeño de este con un caso no utilizado durante el entrenamiento, ante lo cual el modelo es capaz de identificar correctamente la categoría del nuevo objeto. Esta capacidad de predicción ante nuevas muestras es lo que realmente se busca cuando se entrenan modelos de aprendizaje supervisado.

2.3.4. Problemas de clasificación

Es un tipo de problema dentro del enfoque de aprendizaje supervisado en donde para cada muestra del conjunto de datos se tiene como salida una clase o etiqueta a la que pertenece. Por lo que la tarea del modelo es entrenarse para "aprender" a asignar a cada objeto dentro de la categoría correcta a la que pertenece (Steinbach et al., 2006). Desde una vista más técnica, el modelo debe generalizar sobre el comportamiento de los datos e identificar los patrones que definen la pertenencia de una muestra a una clase en particular.

La solución matemática que tiene lugar internamente al finalizar el entrenamiento del modelo de clasificación es la obtención de una función que permite separar de la mejor manera posible los diferentes elementos de cada clase. De manera que sea posible identificar a partir del comportamiento de las variables de entrada a qué clase pertenece una nueva muestra no utilizada durante el entrenamiento.

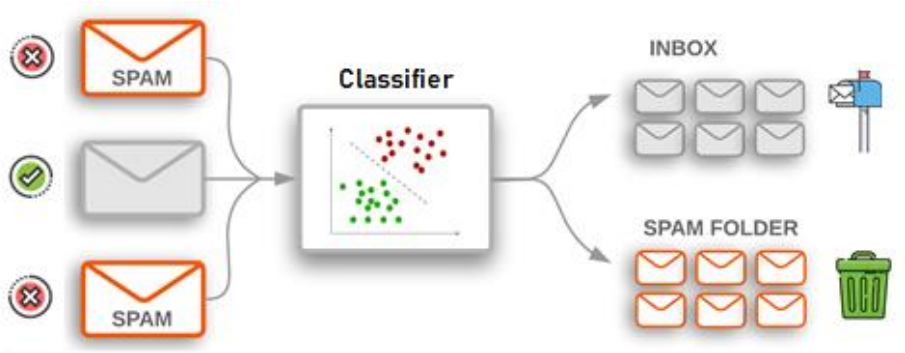

Figura 2-6. Ejemplo ilustrativo de un modelo de clasificación para detección de correos "spam". Fuente: Medium (REF)

En la figura 2-6 se muestra un ejemplo de un modelo de clasificación el cual está orientado hacia la detección de correos "spam", lo que consiste en clasificar los correos que se reciben para diferenciar aquellos que son "spam" de los que no lo son. Internamente el modelo es un clasificador cuyo entrenamiento permitió obtener la función (línea punteada) encargada de separar los correos "spam" (rojos) de los que no son "spam" (verdes), para finalmente enviar cada tipo de correo a la carpeta que corresponden.

3. Antecedentes

La utilización en conjunto de herramientas de procesamiento de lenguaje natural y de minería de datos para extraer información de documentos que contienen texto libre no es un enfoque reciente, de hecho, ha estado presente desde el inicio de muchos servicios que han surgido en esta era digital. Ejemplos clásicos de lo anterior son la funcionalidad de detección de correos "spam" en las casillas de correo electrónico (Radulescu et al., 2014) y la calificación de servicios en función de las opiniones y comentarios que publican los usuarios en algún foro.

Partiendo desde una perspectiva general, (K. Dalal & A. Zaveri, 2011) propusieron una metodología genérica para la clasificación de documentos de texto, la cual inicia con la etapa de pre-procesamiento de los datos y comprende los pasos de eliminación de "stop-words", stemming y determinación de frases, entre otros. Continúan con la etapa de selección de "features", dentro de la cual realizan la identificación de palabras importantes, frases recurrentes e interacción entre "tokens"; para esta etapa los autores sugieren utilizar métodos tales como frecuencia de término – frecuencia inversa de documento (TF-IDF) o indexación semántica latente (LSI). Finalmente proponen la etapa de clasificación automática de texto, en la cual se da paso a la selección de la técnica de aprendizaje de máquina a utilizar para la obtención del modelo.

Tomando como base la metodología anterior, diversos autores han trabajado y realizado sus correspondientes aportes en el tema. Con el propósito de acotar la búsqueda de información, se tomaron como referencia trabajos de clasificación de texto orientados hacia la categorización de productos en plataformas de comercios electrónicos. Iniciando con (Huang et al., 2012), quienes propusieron un modelo de extracción de "features" y categorización de productos a partir de los comentarios de calificación que publican los usuarios en la plataforma; esto lo realizaron utilizando un enfoque de etiquetado secuencial a partir de un esquema de aprendizaje discriminativo basado en "Conditional Random Fields". (Shen et al., 2012) abordaron el problema utilizando un enfoque basado en el sonido de las palabras. Los autores formularon la tarea de categorización como un problema de clasificación bayesiana con dos etapas de aprendizaje y utilizaron algoritmos de suavizado del sonido (fonética) de las palabras para modelar los elementos del lenguaje plasmados como texto en los comentarios de los usuarios.

En (Kozareva, 2015) se aborda el problema desde una perspectiva diferente a la utilizada en los trabajos presentados anteriormente, esto puesto que el autor utiliza como entrada para la extracción de "features" únicamente el texto correspondiente al título de la publicación realizada por el vendedor en la plataforma. El modelado del texto se realizó utilizando unigramas y bigramas a partir de las palabras originales de

los títulos, adicionalmente se incorporó un diccionario que medía el grado de información mutua (correspondencia) entre cada n-grama y la categoría a la que pertenece el producto. La clasificación fue realizada utilizando un modelo multi-clase de regresión lineal generalizada (GLM).

(Cevahir & Murakami, 2016) trabajaron desde una perspectiva que engloba las dos estrategias propuestas en los trabajos anteriores, es decir, utilizaron tanto el texto de los títulos como de los comentarios y revisiones de los usuarios del sitio. En esta propuesta los autores utilizaron técnicas de "tokenizado" diseñadas particularmente para el japonés, idioma en que estaba el texto del dataset trabajado. La clasificación fue realizada mediante "ensemble" de modelos de "deep learning" y K – vecinos cercanos (KNN), en donde el resultado de la predicción se obtuvo promediando la salida de ambas técnicas.

4. Definición del problema

Actualmente los e-commerces tienen dificultades para la revisión y validación de la categorización de los productos que se ofrecen en sus plataformas. Esta tarea que es realizada inicialmente por los vendedores debe ser **siempre** revisada y validada por el equipo del sitio. Sin embargo, debido a la gran cantidad de publicaciones de productos que se crean día a día, los e-commerces no cuentan con el recurso humano suficiente para realizar una revisión y, de ser necesario, corrección manual de la categorización de productos. Sumado a esto, la automatización del proceso no resulta sencilla debido a la subjetividad del uso de lenguaje natural para la descripción de los productos en las publicaciones.

5. Justificación del estudio

Desde el contexto de la problemática de estudio del proyecto, la creciente demanda de los diferentes servicios ofrecidos a través de internet, especialmente de aquellos que prestan las plataformas de comercio electrónico (e-commerce); ha traído consigo un aumento de los sitios (plataformas) que ofrecen este servicio de compra y venta en línea. Debido a esto, las empresas pertenecientes a este rubro se están enfocando cada vez más en el desarrollo de características que permitan mejorar y facilitar la experiencia del usuario para así ser más atractivas y sobresalir ante la competencia.

Es en este punto donde se identificó que existe lugar para aportar en el desarrollo de soluciones desde el enfoque de la ciencia de datos que les permitan a este tipo de empresas mejorar funcionalidades críticas e importantes, como lo es la **categorización de los productos publicados**; razón por la que se decidió plantear y encaminar el proyecto en esa dirección. La idea que sustenta la decisión de automatizar la categorización de los productos publicados es la necesidad que tienen los e-commerces de realizar esta tarea con la mayor precisión posible, esto debido a que algunas de las funcionalidades que impactan directamente en la experiencia del usuario dependen de dicha tarea. Sin embargo, debido a la creciente utilización de sus plataformas, los e-commerces no cuentan con el recurso humano suficiente para validar manualmente la correcta categorización de los productos y se ven fácilmente superados por los cientos de miles de publicaciones que se crean a diario en sus sitios.

La situación descrita anteriormente obliga a proponer soluciones enfocadas en la automatización del proceso, sin embargo, un inconveniente que surge es la subjetividad que se logra filtrar debido al uso de lenguaje natural en los títulos y descripciones de los productos en las publicaciones; lo que es difícilmente superable utilizando únicamente un motor de reglas de decisión. Esto abre una ventana de oportunidad única que justifica el planteamiento de soluciones de automatización desde el enfoque del procesamiento de lenguaje natural (NLP) en conjunto con aprendizaje de máquina, tal como la que se propone en este proyecto.

6. Alcances del trabajo y limitaciones

Se definieron los alcances y limitaciones del proyecto de la siguiente forma:

6.1. Alcances

1. El desarrollo del proyecto se realizará utilizando como base un conjunto de datos de publicaciones de un e-commerce, por lo que el modelo resultante estará orientado a este escenario en particular.
2. Se definirá una metodología para el pre-procesamiento de los datos de entrada utilizando NLP, la cual estará dirigida a aumentar el desempeño del modelo utilizado para realizar la categorización de los productos ofrecidos en las publicaciones.
3. Se obtendrá un modelo de aprendizaje de máquina especializado en la categorización de productos publicados en la plataforma de un e-commerce.
4. El modelo obtenido estará implementado sobre una arquitectura de "Big Data", es decir, utilizando tecnologías de procesamiento distribuido.

6.2. Limitaciones

1. La metodología de análisis mediante NLP sólo se implementará para los idiomas español y portugués.
2. La extracción de "features" mediante NLP se realizará tomando como variables de entrada únicamente el título e idioma de las publicaciones del conjunto de datos del e-commerce.
3. Los recursos de la arquitectura de Big-Data serán seleccionados acordes al tamaño del conjunto de datos del caso de estudio, que es de 20 millones de publicaciones, sin embargo, este tipo de tecnologías son fácilmente escalables para aplicaciones con mayor volumen de datos.

7. Hipótesis de investigación

La hipótesis de investigación planteada es la siguiente:

> "El modelo de aprendizaje de máquina obtenido a partir del conjunto de datos del caso de estudio es una herramienta **adecuada** para la automatización del proceso de categorización de los productos ofrecidos en las publicaciones de la plataforma (e-commerce)"

8. Objetivos

8.1. Objetivo general:

- Desarrollar e implementar un modelo predictivo basado en NLP y aprendizaje de máquina para la categorización de productos ofrecidos en la plataforma del e-commerce.

8.2. Objetivos específicos:

- Realizar el pre-procesamiento de los datos de entrada utilizando técnicas de procesamiento de lenguaje natural (NLP) para identificar y extraer a partir del título los “features” que caracterizan los productos ofrecidos, de manera que sea posible utilizar tales “features” para entrenar un modelo capaz de realizar la categorización de productos.
- Diseñar e implementar un modelo de aprendizaje de máquina para realizar de manera automática la categorización de productos a partir de los “features” previamente caracterizados.
- Evaluar el desempeño del modelo propuesto como una herramienta para la clasificación automática de productos en la plataforma del e-commerce.

9. Tecnologías y herramientas

9.1. Consideraciones técnicas

A lo largo del documento se han ido dando detalles que permiten dimensionar la magnitud y complejidad del problema que se está abordando, algunos de estos están relacionados directamente con la naturaleza propia de la problemática en cuestión mientras que otros radican en la estructura y tamaño de los datos del caso de estudio. Sin embargo, todos contribuyen a conformar un desafío para el cual se debe desarrollar una solución que se encuentre a la altura, no solo en el desempeño del modelo de categorización sino en el cumplimiento de los requerimientos técnicos y funcionales que un problema con estas características supone.

A continuación, se presentan las principales consideraciones que se deben tener en cuenta para el diseño de la arquitectura sobre la cual correrá la solución propuesta:

9.1.1. Cantidad de registros del conjunto de datos

El primer aspecto a considerar radica en el tamaño del conjunto de datos, el cual consta de veinte (20) millones de registros distribuidos en partes iguales para cada uno de los dos idiomas disponibles; lo que quiere decir que se tienen diez (10) millones de registros para español y otros diez (10) millones para portugués. En este momento se hace necesario acotar que cada registro corresponde a una publicación realizada en la plataforma del e-commerce.

Si bien el tamaño mencionado no hace obligatorio el uso de herramientas de Big Data, analizar un conjunto de datos de esta dimensión ya empieza a rayar sobre los límites de capacidad, en términos de hardware, que las computadoras personales comúnmente ofrecen. Además, en una sección anterior se mencionó que, aunque el caso de estudio se basa en un dataset que corresponde a una muestra, la solución se diseñó con la idea de poder ser escalada y utilizada en el contexto real del problema; razón por la cual se consideraron las herramientas de Big Data.

9.1.2. Dimensionalidad del conjunto de datos

El aspecto del tamaño del dataset se complementa directamente con el segundo aspecto a tener en cuenta, el cual corresponde al tipo de problema que se está abordando, que en este caso y visto desde una perspectiva técnica es una clasificación de textos utilizando NLP. Ya es ampliamente conocido que los problemas de NLP suelen ser altamente demandantes de recursos, esto debido a que la dimensionalidad de los datos procesados termina estando ligada a la cantidad de palabras en el vocabulario del dataset, lo cual siempre acaba siendo del orden de las decenas de miles. Por consiguiente, es posible inferir que un dataset con varios millones de registros dará lugar a cientos de miles de palabras, lo que implica que el tamaño final de los datos a modelar será lo suficientemente grande como para requerir que la solución se soporte

sobre una infraestructura capaz de almacenar, procesar y analizar dicho volumen de información.

9.1.3. Títulos de publicaciones en diferentes idiomas

Otro aspecto importante es la dificultad adicional que supone el hecho de tener títulos de publicaciones en dos (2) idiomas diferentes, español y portugués, puesto que se tiene que proponer una solución de modelado lo suficientemente versátil y con la capacidad para poder analizar ambos a la vez, o bien plantear un esquema en donde los registros de cada idioma se procesen y modelen por separado. Cualquiera de ambas opciones se alinea con lo mencionado anteriormente sobre la necesidad de contar con la infraestructura y herramientas adecuadas para que una solución capaz de realizar alguna de estas tareas pueda funcionar correctamente.

9.1.4. Utilización de modelos de Deep Learning para NLP

Finalmente, desde la perspectiva del modelado se tiene conocimiento que el enfoque más comúnmente utilizado en la actualidad para enfrentar problemas complejos de NLP es el de las técnicas de "Deep Learning". Las cuales debido a su gran poder predictivo y diversas topologías han demostrado tener desempeños considerablemente superiores en este dominio que el resto de técnicas de aprendizaje de máquina; razón por la cual representan una opción que no puede quedar afuera del armado de la solución propuesta. Sin embargo, poder entrenar redes de deep learning viene con un alto costo computacional asociado debido a la complejidad de sus topologías, factor que nuevamente ratifica el hecho de tener que seleccionar correctamente el stack de tecnologías a utilizar para cumplir a cabalidad con los requerimientos técnicos asociados a los aspectos de cantidad de registros, dimensionalidad de los datos y complejidad de modelos previamente descritos.

9.2. Arquitectura de tecnologías y herramientas utilizadas

Una vez descritos los principales aspectos considerados para el diseño de la arquitectura y selección de sus componentes, los cuales soportarán la solución propuesta de manera que esta pueda cumplir con los requerimientos técnicos impuestos por dichos aspectos, se da paso a presentar y describir la arquitectura planteada.

9.2.1. Estructura general de la arquitectura

La arquitectura diseñada utiliza un enfoque enteramente basado en **computación en la nube (cloud computing)**, esto debido a que las plataformas que ofrecen este tipo de servicios cuentan con todos los componentes que se pueden necesitar para desarrollar una solución orientada a "Big Data", lo que facilita notablemente el proceso de desarrollo de la misma. Para el caso particular de la arquitectura planteada se necesitaba contar con al menos un servicio de almacenamiento, un servicio de

computación en “cluster” para procesamiento distribuido y la capacidad de escalar en función de la carga de trabajo; elementos que permiten cumplir con los requerimientos técnicos de la solución (descritos en la sección anterior) y todos pueden encontrarse en los principales proveedores de computación en la nube del mercado.

Desde una perspectiva de alto nivel, la arquitectura diseñada se compone de tres (3) grandes capas, iniciando con la capa de **infraestructura**, la cual está conformada por los diferentes servicios utilizados en la plataforma de computación en la nube; luego se encuentra la capa de **procesamiento distribuido**, la cual como su nombre lo indica está integrada por los diferentes “frameworks” que permiten procesar datos de forma distribuida y escalable. Finalmente se encuentra la capa de **lenguaje y herramientas**, la cual está compuesta por el lenguaje de programación utilizado para el desarrollo de la solución y por todas las herramientas que giran alrededor de dicho lenguaje y que forman parte de la misma tales como librerías y paquetes de funciones.

9.2.2. Descripción detallada de la arquitectura

Habiendo comentado desde una perspectiva general la estructura de la arquitectura, ahora se dará paso a realizar una descripción detallada de la misma en donde se definirán cada una de sus capas y los diferentes elementos que las componen; incluyendo también una breve justificación de las selecciones realizadas.

Capa de infraestructura:

Tal y como se mencionó anteriormente, el desarrollo de la solución se realizó sobre una plataforma de computación en la nube, por lo que esta capa está compuesta por todos los servicios de dicha plataforma utilizados para la implementación de la misma.

La plataforma seleccionada fue **Google Cloud Platform (GCP)**, esto obedeció enteramente a un criterio de costo puesto que esta ofrece un cupo de crédito gratuito de 300 dólares que pueden ser utilizados libremente en cualquiera de los servicios ofrecidos. De esta manera se pudo acceder al desarrollo e implementación de la solución en un entorno “cloud” sin tener que incurrir en ningún tipo de gasto.

Los servicios de GCP que fueron utilizados y, por tanto, conforman la capa de infraestructura de la arquitectura son los siguientes:

- **Cloud Storage:** Es un servicio de almacenamiento de información cuya unidad funcional es el “bucket”, que no es más que un depósito en donde pueden alojarse todo tipo de archivos a los cuales luego puede accederse desde cualquiera de los otros servicios de la plataforma.

  Para el caso puntual de la solución propuesta se utilizó un (1) “bucket”, el cual funcionó como punto centralizado de almacenamiento de los siguientes tipos de archivos:

  - ✓ Archivo con el dataset del caso de estudio.

- ✓ Archivos con datasets preprocesados de los diferentes idiomas.
- ✓ Archivos de seguimiento de experimentos.
- ✓ Archivos con las predicciones generadas por los diferentes modelos.
- ✓ Archivos de configuración de entornos de Python.

- **Compute Engine:** Es el servicio estándar de computación de GCP, el cual ofrece recursos de cómputo bajo demanda mediante virtualización de hardware a través de máquinas virtuales, las cuales la plataforma denomina "instancias de VM".

  Para propósitos del proyecto, este servicio se utilizó como entorno de desarrollo y de pruebas, puesto que una máquina virtual (en adelante VM) funciona como un computador independiente que, dentro de GCP, puede definirse para que tenga especificaciones de hardware (CPU y RAM) tan potentes que serían difícilmente alcanzables en un ordenador físico, las cuales permiten manejar el volumen de datos del caso de estudio sin ningún problema.

  Adicionalmente, si bien el código de la solución fue diseñado para ejecutarse de manera distribuida, utilizar recursos de cómputo para procesamiento distribuido resulta bastante costoso y puede ser causante de un gasto excesivo de crédito en la etapa de desarrollo y pruebas, por lo que las VMs del servicio de Compute Engine resultaron ser una buena alternativa durante esta etapa.

- **Dataproc:** Este es el servicio de procesamiento distribuido de GCP, el cual permite crear y utilizar **clusters**[1] de VMs bajo el framework Hadoop.

  Para el proyecto, los clusters de Dataproc representan la plataforma de Big Data sobre la que se ejecutan todos los códigos que conforman la solución, los cuales están especialmente diseñados para correr de forma distribuida sobre este tipo de tecnología. Por lo que se puede inferir que este es el servicio principal dentro de la capa de infraestructura, puesto que sobre este corre todo el pipeline de la solución, desde la ingesta de los datos, pasando por el preprocesamiento, modelado y finalmente la generación de las predicciones.

Capa de procesamiento distribuido:

En la descripción de la capa de infraestructura se presentó a Dataproc como el servicio que provee la tecnología capaz de realizar tareas de procesamiento distribuido mediante de la utilización de un cluster de VMs, sin embargo, para poder llevar a cabo dichas tareas se necesitan elementos capaces de aprovechar correctamente las

[1] Se define como **cluster** al conjunto de máquinas (computadores/VMs) independientes que se encuentran conectadas entre sí y operan en conjunto como si fueran una única gran máquina, a esto se le conoce como procesamiento distribuido.

capacidades de este tipo de tecnología. Es en este punto donde se hace necesario introducir a los motores o "frameworks" de procesamiento distribuido, que son efectivamente los encargados de ejecutar códigos de manera distribuida sobre estas tecnologías.

Antes de describir el motor utilizado resulta pertinente introducir a **Apache Hadoop**, el cual también es en sí mismo un framework que se encarga de conectar, comunicar, configurar y administrar las máquinas que conforman un cluster, permitiéndoles compartir recursos para operar en conjunto como si fueran una sola máquina. Hadoop es la base sobre la cual corren la mayoría de herramientas de Big Data que existen hoy en día, entre esas **Apache Spark**, que es el motor de procesamiento distribuido utilizado en la solución propuesta y el cual se describe a continuación:

- **Apache Spark:** Tal y como se mencionó recién, Spark es un motor de procesamiento que permite ejecutar código de forma distribuida sobre un cluster de máquinas de Hadoop. Es conocido por ser muy rápido y eficiente debido a que realiza todas sus operaciones en memoria principal, razón por la cual ha sido ampliamente adoptado en aplicaciones relacionadas con procesamiento y análisis de grandes volúmenes de datos, tal y como lo es la obtención de modelos de ML.

  Todo el código de la solución está diseñado para correr utilizando Spark, por lo cual, desde la perspectiva del software este framework representa la piedra angular para la ejecución de la misma.

Capa de código y herramientas:

Esta capa la define principalmente el lenguaje de programación elegido para desarrollar el código de la solución, que en este caso particular fue **Python**. Esto claramente debido a que Python es el lenguaje más comúnmente utilizado hoy en día para todo lo relacionado con Data Science, lo cual ha permitido que se haya desarrollado todo un ecosistema de frameworks, librerías y herramientas a su alrededor que facilitan la implementación de soluciones en este campo.

En línea con lo anterior, durante el desarrollo del código de la solución se utilizaron varias de las librerías que componen el "stack" de Data Science de Python, entre las cuales están algunas de las más ampliamente conocidas tales como **Numpy**, **Pandas** y **Scikit-Learn**, entre otras. Sin embargo, estas librerías son nativas de Python y, tal y como se ha mencionado anteriormente, la solución debía ser implementada para que corriera de manera distribuída sobre Spark, razón por la cual estas solo fueron utilizadas como auxiliares para tareas muy específicas al interior de la misma.

Con respecto a los códigos que conforman la estructura nuclear de la solución, es decir, aquellos encargados de las tareas de procesamiento y modelado, estos fueron escritos utilizando principalmente **PySpark**, que es la API de Spark para Python; así como

también otros frameworks para Data Science y ML nativos de Spark. El conjunto de todos los frameworks recién mencionados se describe detalladamente a continuación:

- **PySpark MLlib:** Este es el módulo de PySpark que ofrece todas las funcionalidades de Spark relacionadas con la obtención de modelos de ML, incluyendo desde operaciones de preprocesamiento de datos tales como limpieza y transformación de "features", hasta los diferentes pasos que conforman un "pipeline" de modelado con ML, dentro de los que destacan el entrenamiento, validación, selección y evaluación de diferentes tipos de técnicas.

  Diferentes técnicas (algoritmos) de ML ofrecidas por este módulo fueron probadas como parte del proceso de búsqueda del mejor modelo a utilizar en la solución.

- **Spark-NLP:** Es una librería nativa de Spark exclusivamente orientada a NLP, la cual ofrece todo un espectro de funcionalidades que pueden ser útiles al momento de abordar problemas de este tipo; dentro de las cuales resaltan las múltiples opciones de preprocesamiento y vectorización de texto, así como la posibilidad de utilizar modelos lingüísticos pre-entrenados para tareas comunes de NLP.

  Al interior de la solución esta librería se utilizó para realizar todo el preprocesamiento del texto de los títulos de las publicaciones para ambos idiomas.

- **Elephas:** Si bien se mencionó anteriormente que los modelos de Deep Learning (DL) representan el estado del arte para abordar problemas de NLP, existía una limitante técnica en cuanto a la implementación debido a que el framework más utilizado para desarrollar este tipo de modelos es **Keras**, el cual no corre sobre Spark. Por tanto, para poder utilizar modelos de DL como parte de la solución se recurrió a Elephas, que es un framework que se encarga de escalar modelos de DL de Keras para que puedan correr de forma distribuida sobre Spark.

Resulta importante aclarar que, si bien los frameworks y librerías descritas arriba son nativas de Spark, estas fueron utilizadas desde Python a través de sus correspondientes APIs para este lenguaje.

9.3. Diagrama de la arquitectura de la solución

Habiendo presentado y descrito con un alto nivel de detalle cada una de las capas, y sus respectivos componentes, que conforman la arquitectura sobre la que se soporta la solución propuesta, se presentará a continuación un gráfico en donde se muestra de forma resumida la estructura de dicha arquitectura; en el cual se puede observar también la jerarquía de los diferentes niveles que la componen.

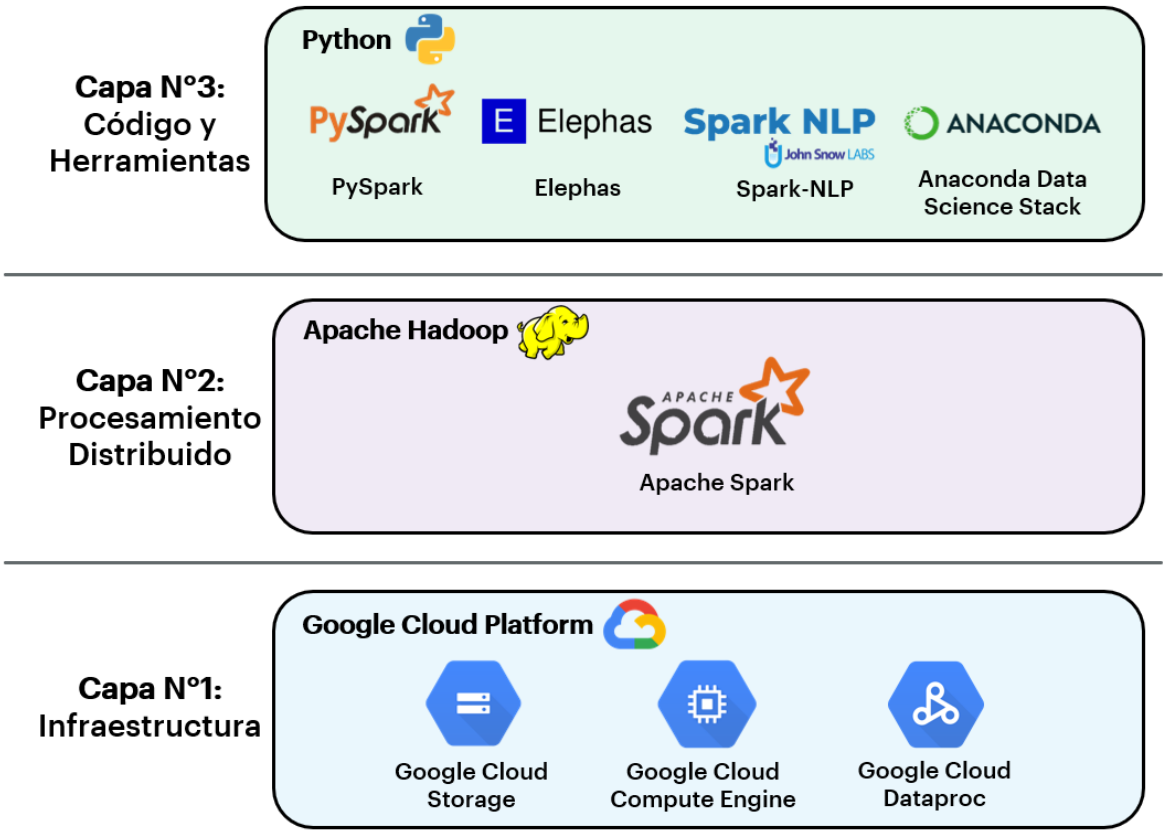

Figura 9-1. Estructura de la arquitectura que soporta la solución propuesta.

10. Metodología de solución propuesta

10.1. El enfoque CRISP-DM

Proponer una solución a una problemática como la que se está abordando, con las particularidades propias de un problema de NLP, la necesidad de un buen desempeño del modelo de ML debido al impacto que la categorización tiene en los servicios de la plataforma y por consiguiente, en la experiencia del usuario; y los requerimientos técnicos debido al gran volumen de datos no resulta ser una tarea sencilla. De hecho, no es posible hacerlo de manera directa siguiendo un único camino debido a la naturaleza experimental e iterativa del ML.

Como consecuencia de lo anterior, se decidió seguir una de las metodologías más comúnmente utilizadas para abordar problemas basados en datos mediante ML, la cual es **Cross Industry Standard Process for Data Mining (CRISP-DM)** (Gallardo, 2009), metodología que propone la serie de pasos fundamentales que deben seguirse para resolver este tipo de problemas, indicando las relaciones existentes entre tales pasos para considerar la experimentación antes mencionada.

En la figura 10-1 se presenta la estructura general de CRISP-DM, los diferentes pasos que la componen y el flujo del proceso propuesto por esta. Se puede ver que no siempre las relaciones son directas sino que entre algunos de los pasos existe bidireccionalidad, lo cual permite reflejar la iteración en la definición del problema y la experimentación durante la obtención del modelo de ML. Adicionalmente, se puede evidenciar que el proceso se plantea de forma cíclica, lo que permite volver al inicio si el mejor modelo obtenido no cumple con las expectativas técnicas o de negocio.

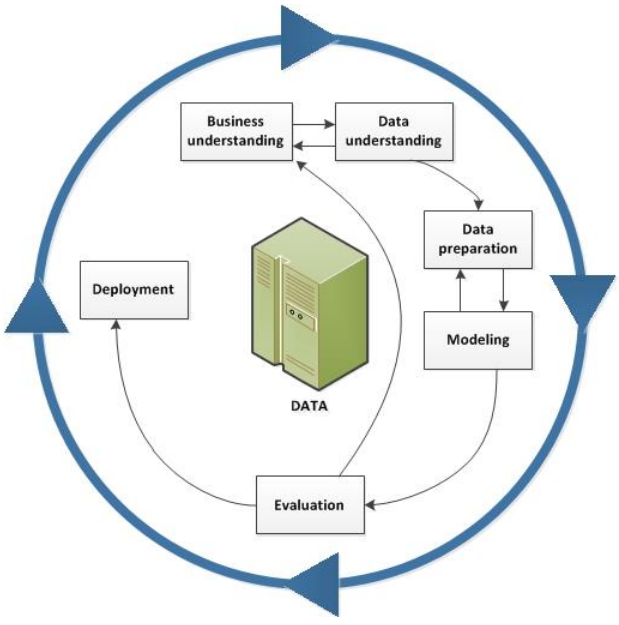

Figura 10-1. Proceso planteado por la metodología CRISP-DM. Fuente: Medium (REF)

A continuación, se describen brevemente cada uno de los pasos que componen el enfoque CRISP-DM, esto a fin de entender la metodología propuesta para la solución de la problemática de interés de este trabajo dado que esta se encuentra basada en dicho enfoque.

1. **Entendimiento del negocio:** Consiste en entender el contexto, los objetivos y requerimientos del problema desde la perspectiva del negocio. Este paso no fue tenido en cuenta debido a la naturaleza académica del problema que se está abordando.
2. **Entendimiento de los datos:** Este paso se enfoca en la exploración, apropiación y entendimiento de los datos con los que se cuenta para la solución del problema. Tiene como finalidad describir el comportamiento, encontrar relaciones, plantear hipótesis y evaluar la calidad de los datos.
3. **Preparación de los datos:** Comprende la aplicación de todas las operaciones de preprocesamiento y transformación requeridas para que los datos puedan ser correctamente utilizados como insumo de entrada al paso de modelado.
4. **Modelado:** Es el paso donde se consumen los datos para realizar el análisis correspondiente mediante la utilización de técnicas de modelado basadas en ML. La idea es obtener un modelo que sea capaz de representar adecuadamente el proceso o fenómeno que subyace a la problemática de estudio, de manera que este pueda utilizarse como solución a dicha problemática y permita mejorar la toma de decisiones e impactar positivamente en el contexto de esta.

   Debe tenerse presente que la obtención del modelo es el resultado de un proceso de experimentación, lo que quiere decir que para llegar al modelo adecuado deben probarse diferentes técnicas (algoritmos) de ML, escenarios de variables y maneras de abordar el problema; de ahí la bidireccionalidad entre este paso y el anterior.
5. **Evaluación:** Está orientado a evaluar el cumplimiento de los objetivos de negocio cuando se utiliza el modelo obtenido como una herramienta para la solución de la problemática de estudio. No se debe confundir con la evaluación técnica que se realiza en el paso 4 como parte del proceso de selección del modelo adecuado.

   Claramente, este paso tampoco fue aplicado por tratarse de un proyecto puramente académico.
6. **Implementación:** En este paso tienen lugar el despliegue y puesta en funcionamiento de la solución (modelo + misceláneos[2]) en un entorno

[2] En este contexto este término hace referencia a cualquier código auxiliar que sea requerido para el correcto funcionamiento del modelo.

productivo o en vivo. Por la misma razón dada en los pasos 1 y 5, este paso tampoco fue incluido en la metodología propuesta.

10.2. Enfoque propuesto

Habiendo descrito la metodología CRISP-DM, la cual se mencionó que fue utilizada como base para la elaboración de la metodología de solución que se propone en este trabajo, ahora se dará paso a presentar propiamente el enfoque de solución planteado; el cual se muestra en la figura 10-2.

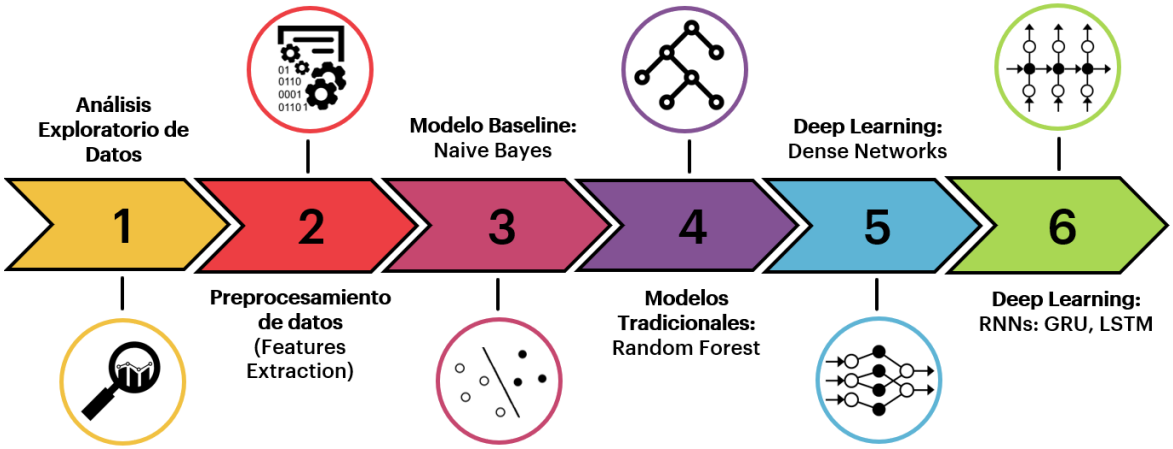

Figura 10-2. Enfoque metodológico propuesto.

En la figura 10-2 puede verse que la metodología de solución planteada se compone de seis (6) pasos, los cuales son los siguientes:

1. **Análisis exploratorio de datos:** Tal y como el nombre lo indica, en este paso se realiza la exploración de los datos para conocer sus principales características y comportamientos. La idea es poder realizar hallazgos que puedan traducirse en decisiones durante el modelado. Este paso se alinea con el paso N°2 de la metodología CRISP-DM.

2. **Preprocesamiento de datos:** Si bien cualquier flujo de trabajo relacionado con ML involucra un paso de preprocesamiento, para el caso de problemas de NLP este cobra una importancia adicional puesto que es necesario preparar y codificar el texto para poder obtener las "features" que serán utilizadas en el modelado; esto es lo que se conoce como extracción de features y es la función principal de este paso.

   Debido a que en la metodología se propone experimentar con diferentes tipos de técnicas de ML (ver figura 10-2, pasos 3 a 6), se hace necesario aplicar también diferentes enfoques de preprocesamiento de texto para realizar la extracción de features; esto debido a que modelos más avanzados pueden

beneficiarse de métodos más complejos de codificación de texto. Todos los métodos de codificación requeridos son aplicados en este paso.

3. **Modelo Baseline:** Proponer una solución con el nivel de complejidad que viene asociada a la utilización de modelos de ML conlleva a la necesidad de obtener beneficios a cambio de dicha complejidad. Por lo cual, se espera que cuanto más complejo sea el modelo, mejor desempeño tenga y por consiguiente, mayores beneficios aporte a la solución del problema.

   Una manera de medir la mejora que se obtiene a medida que se complejiza el modelo utilizado es fijando un punto de partida por medio de la utilización de un modelo inicial o de "baseline", el cual debe ser lo suficientemente sencillo como para representar el mínimo esfuerzo requerido para dar una solución al problema. Como consecuencia de lo anterior, se espera que con cada paso adicional de complejidad se mejore el desempeño del modelo en relación a dicha complejidad y tomando como referencia el modelo "baseline". En caso de no ocurrir, es decir, cuando la complejidad requerida es mayor que la mejora en desempeño, entonces se descarta el modelo probado.

   El algoritmo baseline utilizado como referencia para el modelado fue **Naive Bayes**, que si bien es una de las técnicas más sencillas de ML, representa un buen punto de partida puesto que es conocido por desempeñarse bien en problemas de NLP.

4. **Modelo Tradicional:** Luego de iniciar con el modelo baseline se propone avanzar en complejidad utilizando una técnica de ML tradicional como lo es **Random Forest**, la cual suele ofrecer buenos desempeños en problemas complejos debido a su característica de ser un ensamble de múltiples árboles de decisión. Si bien los modelos tradicionales no son la opción por defecto para NLP, estos representan una buena alternativa a considerar para cualquier problema, razón por la cual fueron tenidos en cuenta.

5. **Deep Learning – Redes Densas:** En este paso de la metodología se propone utilizar redes neuronales de Deep Learning, las cuales representan el estado del arte y opción por defecto para problemas de NLP debido a los buenos resultados que ofrecen para este campo de aplicación. Si bien en la actualidad existen topologías más complejas y avanzadas, en este paso se utiliza la topología clásica que corresponde a las denominadas redes densas o completamente conectadas; las cuales representan el punto de partida para la experimentación con Deep Learning.

   Aunque esta topología es la opción "sencilla" dentro del dominio de DL, cuenta con un alto poder predictivo debido a la complejidad de las conexiones que surgen como producto de la profundidad de la red, lo cual la hace una opción a

considerar como parte de la metodología. Un aspecto a favor es que, aún cuando las redes de DL demandan una cantidad elevada de recursos para su entrenamiento, esta topología resulta ser la menos demandante por ser la menos compleja de todas; hecho que resulta positivo de cara al consumo de recursos en la plataforma de computación en la nube (GCP).

6. **Deep Learning – Redes Recurrentes (LSTM y GRU):** Como paso final de la metodología propuesta se experimentará con la topología de DL denominada redes recurrentes (RNN), las cuales están especialmente diseñadas para abordar problemas en los que existe secuencialidad entre los elementos que conforman cada registro del dataset; ejemplos de este tipo de problemas son aquellos relacionados con series de tiempo debido al orden temporal de las observaciones y aquellos de NLP como producto de la secuencialidad propia del lenguaje.

   Este tipo de redes neuronales pueden "entender" la dependencia o relación que existe en el orden de los elementos de una secuencia (registro) debido a un mecanismo de memoria, en donde la red procesa dichos elementos de uno a la vez y considera no solo el efecto del elemento actual sino también de los que vinieron antes. Las configuraciones más comunes de RNNs son **Long Short Term Memory (LSTM)** y **Gated Recurrent Unit (GRU)**, de las cuales se probará la LSTM.

   Un detalle importante a considerar es que esta topología resulta ser **mucho** más demandante de recursos computacionales que las redes densas, esto debido al hecho de que los registros se procesan elemento por elemento para considerar la secuencialidad. Hecho sobre el que debe prestarse especial atención para no incurrir en un consumo excesivo de recursos en GCP.

Aunque la metodología propuesta llega hasta la experimentación con modelos de RNNs, el estado del arte actual en NLP se ubica sobre las denominadas **redes de transformadores**, las cuales no fueron consideradas debido a la dificultad de utilizarlas de manera distribuida sobre Spark al momento de desarrollar este trabajo (2020). Sin embargo, antes de esta nueva topología las redes recurrentes eran el estándar de facto para NLP, por lo que se espera obtener buenos resultados con la metodología tal cual se plantea en la figura 10-2.

Finalmente, si bien con el enfoque propuesto se espera una mejora incremental con cada paso debido a la utilización de modelos más avanzados y complejos, la definición del modelo final a utilizar para dar solución a la problemática se realizará no solo considerando el valor de la métrica de desempeño elegida sino también la relación costo computacional – beneficio obtenido de cada modelo.

11. Implementación de la solución

Una vez se ha presentado la metodología que representa el enfoque de solución propuesto, la finalidad de esta sección es describir detalladamente el proceso de desarrollo y las características que definen a cada una de las piezas que se implementaron para la materialización de dicha metodología. Para esto se visitarán cada uno de los pasos previamente descritos en la sección 10.2, sin embargo, en esta ocasión se dan los detalles y por menores técnicos de la implementación de cada uno de estos.

1. **Análisis exploratorio de datos:** Con el objetivo de tener un primer acercamiento a los datos del caso de estudio, los pasos de exploración que se siguieron estaban orientados a conocer la estructura, propiedades y comportamientos de la información contenida en el dataset. Estos pasos fueron los siguientes:
    - Obtención del número de filas y columnas del dataset.
    - Impresión de una muestra de los datos para tener una noción de su aspecto.
    - Exploración de la estructura del dataset, esto es, obtener los nombres de las columnas que lo componen y los tipos de datos de cada una de estas.
    - Obtención de la cantidad de registros (filas) por idioma.
    - Análisis en función de la columna **"label_quality"**.
    - Obtención de la cantidad de categorías existentes a nivel general y de la cantidad de registros en cada una de estas.
    - Análisis de la distribución de la cantidad de registros por categoría según el idioma.
    - Identificación de categorías únicas de cada idioma, esto hace referencia a las categorías que están presentes en un idioma pero no en el otro.
    - Análisis inicial de la distribución de la cantidad de palabras (tokens) en los registros del dataset.

    Los resultados obtenidos luego de realizar cada uno de los pasos que componen el análisis exploratorio descrito se presentan en la siguiente sección (sección de resultados).

    Para la ejecución del código desarrollado para el EDA se utilizó una VM del servicio **"Compute Engine"** de GCP compuesta por 8 vcpus y 64 Gb de RAM. Debido a que en este paso solo se realiza un análisis exploratorio, no fue

necesario utilizar ninguna plataforma de procesamiento distribuido, por lo que el código fue escrito y ejecutado enteramente en Python.

2. **Preprocesamiento de datos:** Siempre que se aborda un problema de NLP, la etapa de preprocesamiento resulta ser una combinación secuencial de dos pasos diferentes. El primero está relacionado con el hecho de que el texto que se recibe a la entrada se encuentra en formato "libre", es decir, no tiene ninguna estructura específica más allá de la estructura natural propia del lenguaje. Esto supone un impedimento para cualquier análisis debido a la complejidad de los lenguajes naturales, en donde una misma palabra puede escribirse de varias formas, existen signos de puntuación, se utilizan mayúsculas y minúsculas, entre otras muchas cosas.

   Es por lo anterior que el primer paso corresponde a una **preparación** del texto, en donde lo que se busca es simplificarlo para poder facilitar la extracción de features que ocurrirá más adelante. Dicha simplificación tiene lugar mediante la aplicación de una serie de operaciones sobre el texto, las cuales para el caso particular de la solución propuesta fueron:

   a. Conversión de todo el texto de los títulos a minúsculas.

   b. Reemplazo de todos los caracteres numéricos por un "placeholder" común.

   c. Eliminación de caracteres especiales tales como ",", ";", ".", ":", "#", etc.

   d. Reducción de múltiples caracteres de espacio (" ") consecutivos a uno solo.

   e. Tokenización, que consiste en descomponer el texto de cada título en las palabras que lo conforman.

   f. Eliminación de "stop-words". En donde las "stop-words" son aquellas palabras auxiliares cuya función suele ser conectar ideas, por lo que carecen de significado útil para el análisis realizado.

   Como resultado del paso de preparación se obtiene para cada título de publicación una lista con las palabras (tokens) que lo conforman, las cuales ya se encuentran modificadas por todas las operaciones antes mencionadas.

   En este punto se aplica el segundo paso, el cual consiste en la generación de **"N-gramas"** para cada título de publicación a partir de los tokens obtenidos del paso anterior. Los "N-gramas" son tokens de nivel superior que están compuestos por N tokens simples consecutivos, en donde N va desde 1 hasta el número que se desee, que en la práctica no suele ser superior a 3. Estos tienen la finalidad de agregar un poco de contexto en función del orden de los tokens. Los N-gramas generados para la solución fueron:

a. **Mono-gramas:** Los cuales son N-gramas compuestos por un solo token simple, es decir, una palabra.

b. **Bi-gramas:** Que corresponden a tokens compuestos por dos (2) tokens simples consecutivos, es decir, una asociación de dos palabras separadas por un espacio como un mismo token. La generación de bi-gramas es abarcativa, lo que quiere decir que además de estos también se obtienen todos los mono-gramas que los conforman.

En general, los N-gramas suelen ser el insumo de texto preprocesado que se ingresa como entrada a la parte de **vectorización** del texto, la cual consiste en la generación de las "features" que conformarán el dataset que será utilizado para obtener los modelos. Debido a que en la metodología propuesta se plantea la utilización de diversos tipos de técnicas de ML, existe la posibilidad de que algunas de estas se beneficien de diferentes maneras de vectorizar los N-gramas; razón por la cual se decidió realizar de la obtención de los mono y bi-gramas mencionados de forma separada, esto a fin de poder realizar la vectorización de manera independiente según lo requiera cada técnica.

Como consecuencia de lo anterior, el código de preprocesamiento de la solución se diseñó para que por defecto ejecutara ambas opciones de generación de N-gramas por separado, obteniendo así dos versiones preprocesadas del dataset del caso de estudio; una en donde para cada título de publicación se tienen los diferentes mono-gramas que lo componen y otra en donde para cada título se tienen el conjunto de mono y bi-gramas que resultan de este.

Un detalle importante es que todo el procedimiento descrito hasta ahora en este paso N°2 se aplica de forma independiente para cada idioma. Esto se garantiza dividiendo el dataset del caso de estudio en dos datasets al inicio de este paso, uno para todas las publicaciones con títulos en español y otro para todas las publicaciones con títulos en portugués.

De lo anterior se obtiene que el resultado final de este paso son cuatro (4) datasets preprocesados, así:

- Dataset con los mono-gramas de las publicaciones en español.
- Dataset con los mono y bi-gramas de las publicaciones en español.
- Dataset con los mono-gramas de las publicaciones en portugués.
- Dataset con los mono y bi-gramas de las publicaciones en portugués.

Finalmente, con respecto a la plataforma de computación utilizada para llevar a cabo este paso, dado que este representa el primer paso orientado a la obtención de la solución del problema y en el cual se empieza a procesar el dataset del caso de estudio; en este si se utilizó una plataforma de procesamiento distribuido

definida por un cluster de máquinas de Dataproc en GCP. Dicho cluster se definió con las siguientes especificaciones técnicas:

- 1 máquina o nodo "driver" con 4 vcpus y 16 Gb de RAM.
- 2 máquinas o nodos "workers" con 8 vcpus y 64 Gb de RAM cada uno.

Adicionalmente, con respecto al código desarrollado es necesario resaltar que todo el proceso de preprocesamiento descrito tuvo lugar en Spark mediante la utilización de la librería Spark-NLP.

Los cuatro (4) datasets preprocesados resultantes se almacenan en el bucket de GCP descrito en 9.2.2, lugar desde donde serán consultados durante los pasos relacionados con el proceso de modelado.

3. **Modelo Baseline – Naive Bayes:** En la sección anterior se mencionó la necesidad que existe de empezar por un modelo "baseline" cuando se solucionan problemas basados en datos utilizando modelos de ML. En donde para la solución propuesta se planteó la utilización del algoritmo **Naive Bayes** como punto de partida para el proceso de modelado, el cual además de ser sencillo resulta bastante bueno para problemas de NLP debido a su funcionamiento basado en probabilidades condicionales.

   Lo anterior se explica con el siguiente planteamiento. Dado que se está abordando un problema de clasificación de texto, en donde la idea es asignar cada producto a la categoría a la que pertenece en función del título de su publicación; resulta coherente pensar que existirán palabras (tokens) cuya aparición en un título está directamente ligada a una categoría particular, por ejemplo, es claro que la ocurrencia del token "laptop" en un título va estar relacionado con la categoría "computadores" casi que en el 100% de los casos. Por tanto, la probabilidad condicional para tal categoría dada la ocurrencia de dicho token va a ser alta, hecho que se cumple para muchos otros casos, permitiendo que el algoritmo Naive Bayes presente un buen desempeño en este tipo de problemas.

   Consideraciones de diseño

   - Tal y como se mencionó en algunas partes de este documento, la tarea de asignar cada producto a la categoría a la cual pertenece en función de su título de publicación corresponde a un problema de **clasificación multiclase**. Esto debido a que cada producto puede ser categorizado en una de las cientos de categorías que existen en el dataset del caso de estudio. Por lo tanto, todos los algoritmos y métricas de desempeño utilizadas en este y en **todos** los otros pasos de modelado para la

implementación de la solución se orientan hacia la resolución de un problema de este tipo.

- Llegado este punto se hace necesario establecer que, tanto para este paso como para **todos** los otros pasos de modelado, se definió como criterio de diseño que se obtendrían modelos independientes para cada idioma. Esto debido a que cada idioma tiene su complejidad propia asociada a su vocabulario particular, las cuales combinadas podrían ser más difíciles de capturar para un único modelo. De manera que si se obtiene un modelo por idioma se le permitiría a estos enfocarse y capturar las particularidades de un único lenguaje, aumentando así las probabilidades de éxito de la solución.

  Como resultado de lo anterior, el modelo final de la solución va a ser realmente una composición de dos modelos diferentes, el mejor modelo que se encuentre para español y el mejor modelo que se encuentre para portugués.

- Alineado con el inciso anterior, otro criterio de diseño que se definió para **todos** los pasos de modelado está relacionado con el enfoque para realizar la evaluación de los modelos obtenidos; el cual se explica a continuación.

  Una de las columnas que viene incluida en el dataset del caso de estudio es **"label_quality"**, para la cual cada registro tiene uno de los siguientes valores "reliable" o "unreliable". La finalidad de esta columna es indicar si la categoría a la cual fue asignado el producto está revisada y validada por parte del equipo del e-commerce y, por tanto, puede asumirse como correcta. Básicamente, los registros cuyo valor sea "reliable" en la columna "label_quality" cumplen con lo mencionado anteriormente, y los registros con valor "unreliable" tienen una categoría asignada pero no existe certeza que esa sea correcta.

  Por lo tanto, teniendo en cuenta que en el proceso de obtención de modelos de ML siempre debe existir un set de **testeo** que debe ser utilizado al final para evaluar y comprobar el correcto desempeño del modelo ante datos nuevos; se definió que dicho set estaría compuesto por todos los registros cuya categoría se conoce que es correcta ("reliable"). Esto con el fin de asegurar que la verificación del poder predictivo real del modelo, es decir, el desempeño sobre datos nuevos, se realiza con ejemplos correctos y que pueden dar una buena estimación de lo que puede esperarse si el modelo llegara a ser servido en producción.

Datos de entrada

Para ambos idiomas, los datasets utilizados como entrada para el paso de obtención de los modelos "baseline" son los que contienen los mono y bi-gramas generados en el paso de preprocesamiento de datos, los cuales son obtenidos desde el bucket en donde fueron almacenados al final de dicho paso.

Extracción de features

En el paso N°2 se mencionó que, generalmente, en los problemas de NLP el paso final de preprocesamiento de los datos consiste en la vectorización de los N-gramas, lo cual no es otra cosa más que extraer a partir de estos las features (columnas) que conformarán el dataset que se utilizará para obtener los modelos de ML.

Existen diversas maneras de llevar a cabo la vectorización requerida, siendo la más básica aquella conocida como "bolsa de palabras", en donde cada N-grama representa una columna del dataset y para cada registro, si un N-grama no está presente en este entonces el valor en la columna correspondiente a dicho N-grama será cero (0) o en caso de que si esté, entonces el valor será la cantidad de veces que el N-grama aparezca en dicho registro; como una especie de codificación "one-hot".

Es posible encontrar algunas maneras de vectorización que, si bien se basan en el método de la bolsa de palabras, tienen en cuenta algunas consideraciones adicionales para bien sea mejorar el desempeño de los modelos o brindar robustez al proceso de vectorización del texto. Tal es el caso de la vectorización por **"hashing"**, que resulta de utilizar una función para mapear los N-gramas de texto a números antes de construir la bolsa de palabras, y cuyo principal beneficio es poder considerar nuevos N-gramas que no estaban presentes en el dataset al momento de entrenar el modelo[3]. También se cuenta con la vectorización por TF-IDF, la cual consiste en construir la bolsa de palabras y luego ponderar los N-gramas para dar mayor peso a aquellos que son menos comunes y dar menor peso a aquellos muy comunes en los registros; esto con la idea de que aquellos muy comunes pueden ser palabras que no aportan para el análisis realizado[4].

Para propósitos de la implementación del paso en cuestión se experimentó con diversos enfoques de vectorización en función de los métodos descritos

[3], [4] Tanto la vectorización por hashing como por TF-IDF son mencionadas pero no explicadas en detalle puesto que no son el foco de interés de este documento. Se propone al lector que indague por su parte en caso de que se encuentre interesado en conocer el funcionamiento de estas técnicas.

anteriormente, de manera que se pudiera encontrar aquel que, para cada idioma, permitiera obtener el mejor modelo posible. Los enfoques que se consideraron fueron:

- Bolsa de palabras + TF-IDF
- Vectorización por hashing + TF-IDF

Claramente, como se obtiene un modelo por idioma, tanto para español como para portugués se probaron ambos enfoques mencionados.

Definición de modelos Naive Bayes

Aunque al inicio de este paso se mencionó muy superficialmente que este algoritmo se basa en la teoría de probabilidad condicional, la finalidad de este documento no es entrar en mayores detalles sobre el funcionamiento del mismo. Por lo que se asume que el lector ya conoce o puede consultar por su cuenta la teoría detrás de esta técnica y se pasará directamente a describir la implementación de la misma como parte de la metodología de solución.

Debido a su sencillez, la definición de un modelo predictivo utilizando el algoritmo de Naive Bayes solo requiere de la definición de un (1) solo hiperparámetro, que es el **factor de suavizado** ("smoothing"). El cual tiene la finalidad de evitar que el valor de la probabilidad condicional resultante se vaya a cero (0) cuando alguno de los features tiene valor cero.

A continuación, se presenta el rango de valores que se consideró para este hiperparámetro como parte del proceso de "tuning" para encontrar el mejor modelo Naive Bayes de cada idioma y enfoque de extracción de features.

Hiperparámetros Naive Bayes	
Hiperparámetro	**Valores**
Smoothing	0 a 1

Tabla 11-1. Hiperparámetros algoritmo Naive Bayes.

Finalmente, solo resta comentar que debido a que este paso se desarrolló en Spark, para la utilización de este algoritmo se apalancó en la implementación del mismo que viene incluida como parte del módulo de ML de Spark, él cual es denominado **Spark MLlib**.

Entrenamiento, validación y evaluación de modelos

Se inicia esta parte comentando que el esquema de entrenamiento y validación elegido para llevar a cabo el tuneo de modelos en los diferentes experimentos, donde un experimento es la combinación de un idioma y un enfoque de extracción de features, fue el llamado **"single validation"** o validación sencilla.

En el cual solo se realiza una única división del conjunto de datos de entrenamiento en la porción a utilizar para entrenar y la porción a utilizar para validar, es decir, no existe validación cruzada (cross-validation).

Esto debido a que para cada idioma se cuenta con un dataset de alrededor de 9.5 millones de registros (excluyendo la porción de testeo), con lo cual se cuenta con un volumen de datos suficiente como para que el entrenamiento resulte representativo y consistente sin necesidad de realizarlo en repetidas ocasiones. De estos 9.5 millones de registros se utiliza el **95%** para entrenamiento y **5%** para validación de los modelos.

Con respecto a la evaluación, se hace necesario indicar que la métrica de desempeño elegida para medir a los modelos fue el **"weighted F1 score"**, la cual es una versión del F1 score adaptada para problemas de clasificación multiclase; en donde se calcula el F1 score del modelo para cada clase y se obtiene el promedio de estos ponderando según el número de observaciones de cada una de estas. De esta manera se tiene en cuenta el hecho de que existirán categorías con muchas más publicaciones que otras.

La métrica de desempeño planteada se utilizará tanto en este como en **todos** los otros pasos de modelado de la metodología. En general, el proceso a seguir es realizar el entrenamiento para los diferentes conjuntos de hiperparámetros bajo el esquema de "single validation", en donde se selecciona el conjunto ganador en función del "weighted F1 score" en el set de validación y finalmente se evalúa el desempeño del modelo definido por dicho conjunto de hiperparámetros calculando su "weighted F1 score" en el set de testeo.

Guardado de resultados

Como última instancia, los resultados obtenidos luego de la ejecución de este paso son guardados en el bucket de Google Cloud Storage para futuros análisis. Esto se hace de la siguiente manera:

- Se guarda un archivo de texto (.csv) que contiene las métricas de desempeño (weighted F1 score) en entrenamiento, validación y testeo del **mejor** modelo Naive Bayes de cada experimento.
- Se guarda un archivo de texto (.csv) con las predicciones de cada uno de los mejores modelos de los diferentes experimentos para el set de testeo.

Implementación

Se utilizó un cluster de máquinas de Dataproc en GCP. Dicho cluster se definió con las siguientes especificaciones técnicas:

- 1 máquina o nodo "driver" con 4 vcpus y 16 Gb de RAM.

- 2 máquinas o nodos “workers” con 8 vcpus y 64 Gb de RAM cada uno.

4. **Modelo tradicional – Random Forest:** Luego de haber descrito anteriormente el proceso de implementación del modelo baseline y, siguiendo con la metodología de solución propuesta, este paso consiste en la descripción del proceso de desarrollo del modelo de ML tradicional elegido, el cual es Random Forest.

   Es necesario traer a colación algo que se mencionó en la sección anterior, y es que los modelos de ML tradicionales no son actualmente la opción por defecto a considerar cuando se abordan problemas de NLP, sin embargo, suelen presentar buenos desempeños indistintamente del problema o la aplicación; razón por la cual se incluyó a Random Forest para representar a esta “familia” de técnicas como parte de los modelos a considerar en la metodología propuesta.

   Consideraciones de diseño

   En este paso se mantienen todas las consideraciones de diseño ya descritas en el paso N°3, razón por la cual no se hace necesario volverlas a presentar aquí.

   Datos de entrada

   Similarmente al paso N°3, en este paso también se utilizan como datos de entrada para la experimentación con modelos Random Forest los datasets que contienen la agrupación de todos los mono y bi-gramas que componen los títulos de las publicaciones de cada idioma.

   Extracción de features

   De la misma manera, la extracción de features para el armado de los datasets a ingresar a los modelos Random Forest se realiza siguiendo los enfoques planteados en el paso anterior, que son:

   - Bolsa de palabras + TF-IDF
   - Vectorización por hashing + TF-IDF

   Definición de modelos Random Forest

   Random Forest (RF) es una técnica de ML tradicional que pertenece a la categoría conocida como “modelos de ensamble”, en donde la característica principal de los algoritmos de esta categoría es que el modelo que se obtiene es en realidad un “meta-modelo” que resulta de la combinación de múltiples modelos más sencillos; los cuales en el caso de Random Forest son **árboles de decisión**.

   El algoritmo de Random Forest y en general todas las técnicas de ensamble de modelos se basan en el postulado de la “sabiduría de las multitudes”, el cual

enuncia que la mejor aproximación de un valor se puede conseguir consensuando la opinión de múltiples “expertos” o “conocedores” del tema en vez de basarse en una única estimación del mismo. En el caso de RF, estos expertos son los múltiples arboles de decisión que son entrenados de manera independiente para que cada uno arroje su propia predicción, las cuales al final son promediadas para obtener la predicción final del modelo. Similarmente al paso anterior, se insta al lector a profundizar por su cuenta sobre los por menores y detalles particulares del algoritmo de Random Forest.

En cuanto a los hiperparámetros que se utilizan para definir un modelo RF, a continuación se describen aquellos que fueron tenidos en cuenta durante la experimentación para tunear dicho algoritmo:

- **Profundidad máxima (maxDepth):** Es la profundidad o cantidad de niveles límite que se le permite crecer a cada uno de los arboles que conforman el ensamble.
- **Mínimo de instancias por nodo (minInstancesPerNode):** Es la cantidad mínima de observaciones que deben resultar en cada nodo hijo al dividir su correspondiente nodo padre, si no se cumple no se sigue dividiendo dicha rama. Aplica para todos los árboles del ensamble.
- **Número de árboles (numTrees):** Es la cantidad de árboles independientes que se generarán para construir el ensamble.
- **Estrategia de muestreo de features (featureSubsetStrategy):** Con la finalidad de introducir varianza en el ensamble se busca que los árboles que lo conforman sean diferentes entre sí, esto se logra utilizando una muestra diferente de features para generar cada uno de estos. Este hiperparámetro indica la cantidad de features a incluir para generar los diferentes árboles del ensamble.
- **Ratio de muestreo (subsamplingRate):** Tiene un propósito similar al anterior pero se orienta a definir la cantidad de observaciones (filas) del dataset que se utilizarán para construir los árboles ensamble. Las cuales son muestreadas aleatoriamente antes de construir cada árbol.

Seguidamente se presenta una tabla en donde se resumen los diferentes rangos u opciones que se consideraron para cada uno de los anteriores hiperparámetros durante la optimización de los mismos a fin de obtener el mejor modelo RF de cada experimento.

Hiperparámetros Random Forest	
Hiperparámetro	**Valores**
maxDepth	[25, 27, 30]
minInstancesPerNode	[1, 2, 5]
numTrees	[200, 350, 500]
featureSubsetStrategy	['sqrt', 'log2']
subsamplingRate	[0.5, 0.6, 0.7, 0.8, 0.9]

Tabla 11-2. Hiperparámetros algoritmo Random Forest.

Es necesario aclarar que la definición de los valores presentados en la tabla 11-2 se realizó mediante ensayo y error, lo que quiere decir que estos fueron los valores a los que se llegó luego de sucesivas pruebas.

En línea con el paso anterior, para este paso también se utilizó la implementación de Random Forest que viene incluida como parte de la librería de ML de Spark, MLlib.

Entrenamiento, validación y evaluación de modelos

El proceso de entrenamiento, validación y evaluación de modelos Random Forest para cada experimento (idioma + extracción de features) se realizó de la misma manera que como se describió en el paso N°3 para los modelos Naive Bayes, por lo que no se hace necesario repetirlo en este paso.

Guardado de resultados

Para esta parte se siguió la misma mecánica que en el paso anterior, en donde se guardan dos (2) archivos de texto (.csv) en el bucket de GCP, uno con las métricas de desempeño del mejor Random Forest de cada experimento y otro con las predicciones de cada uno de estos en el set de testeo.

Implementación

La ejecución de este paso N°4 y todos sus experimentos asociados se realizó utilizando un cluster de Dataproc con las mismas especificaciones que las detalladas en el paso N°3.

5. **Deep Learning – Dense Networks:** En la sección anterior se comentó que las redes de DL representan el estado del arte, en cuanto a modelos de ML se refiere, para abordar problemas de NLP. Así mismo, también se explicó el por qué resulta necesario y beneficioso iniciar la experimentación con la topología clásica y posteriormente agregar complejidad mediante la utilización de otras topologías más avanzadas.

Como consecuencia de lo anterior, se procede directamente a describir detalladamente el proceso de desarrollo e implementación de los modelos de redes densas de DL como parte de la solución propuesta.

Consideraciones de diseño

Alineado con los pasos anteriores de modelado de la metodología, pasos N°3 y N°4, para este paso también se mantienen las mismas consideraciones de diseño que en los anteriores.

Datos de entrada

En este punto se presenta la primera diferencia con la implementación de los pasos anteriores, pues para este paso se utilizan como entrada los datasets que contienen únicamente los mono-gramas que componen los títulos de las publicaciones de cada idioma. La razón de esto se explica en detalle en el siguiente inciso.

Extracción de features

Si bien los enfoques de extracción y vectorización utilizados en los pasos anteriores pueden usarse tranquilamente para alimentar modelos de redes neuronales de DL, estos carecen de la posibilidad de representar en profundidad el significado y contexto de las palabras debido a que no son más que conteos de apariciones de los diferentes tokens.

Es por lo anterior que, debido al alto poder predictivo de los modelos de DL, resulta conveniente utilizar alguna estrategia de vectorización capaz de capturar las interacciones complejas del lenguaje, pues de esta manera se puede aprovechar mejor la capacidad de dichos modelos y obtener mejores resultados. Debido a esta necesidad surgió un nuevo enfoque de extracción y vectorización denominado **"word embeddings"**, que es el que se utilizó para crear los datasets que se ingresan como entrada a los modelos de DL de este paso.

La técnica de "word embeddings" consiste en mapear cada palabra a un vector de números, en donde estos vectores pertenecen a un espacio N dimensional, siendo N la dimensión de los vectores, el cual intenta modelar las interacciones del lenguaje a través de las interacciones entre dichos vectores, en particular, con la distancia entre estos. Con la idea de que si son obtenidos correctamente, estos vectores capturarán el significado y contexto de las palabras del lenguaje y se podría esperar, por ejemplo, que los dos (2) vectores que representen las palabras "perro" y "gato" se encuentren más cercanos entre sí que los vectores que representen las palabras "perro" y "avión".

Existen en la actualidad muchas maneras y enfoques para obtener los "word embeddings" de un dataset, algunas mucho más elaboradas que otras, sin

embargo, para este paso se utilizó una de las formas más simples denominada **"capa inicial de embedding"**, la cual consiste en agregar una capa al inicio de la red neuronal cuya única finalidad es generar y proveer los embeddings al resto de la red. Debe tenerse presente que esta capa actúa como una función de mapeo entre el texto a analizar y la parte de la red encargada del modelado, pues transforma la secuencia de texto a una secuencia de vectores, siendo cada uno de estos vectores el embedding correspondiente de cada palabra de la secuencia de texto. Aunque esta capa no toma parte en el análisis si lo afecta directamente pues este se hace con los embeddings generados por esta, así que los resultados que pueden obtenerse dependen en gran medida de que tan buenos sean estos.

Por tal motivo, se hace necesario mencionar como se generan los embeddings con la capa inicial. En el momento de inicialización del modelo todos los pesos de la red se asignan aleatoriamente, esto incluye a los de la capa de embedding, por tanto, los embeddings iniciales son meramente vectores aleatorios y no representan ninguna característica de las palabras del lenguaje. Sin embargo, al hacer parte de la red, los pesos de la capa inicial también se ajustan iterativamente durante el entrenamiento con "back-propagation". Esto quiere decir que la capa de embedding "aprende" a representar las palabras del texto y las características de estas a la vez que la red intenta mejorar su desempeño para la predicción de la variable objetivo. Por tanto, lo que realmente se obtiene de dicha capa son los embeddings que mejor representan cada palabra según el contexto del problema, no a un nivel lingüístico general.

Definición de modelos de redes densas de DL

Tal y como se mencionó en la sección anterior, las redes densas son la topología clásica de DL y se caracterizan por el hecho de que, dada una capa cualquiera de la red, todas las neuronas (unidades) que componen dicha capa se interconectan con todas las neuronas de la capa siguiente, y así para todas las capas; es por esto que reciben el nombre de redes completamente conectadas.

De la misma manera que en los pasos anteriores, se procederá directamente a describir la implementación de este tipo de modelos y se deja a voluntad del lector la posibilidad de indagar más sobre la teoría y conceptos asociados a redes neuronales.

Para la definición de un modelo basado en una red densa de DL existen dos (2) tipos de hiperparámetros, aquellos orientados a modificar la estructura de la red y aquellos orientados a controlar el entrenamiento de esta; en donde ambos tipos pueden modificar el poder predictivo del modelo. A continuación, se describen los diferentes hiperparámetros, según su tipo, que fueron tenidos en cuenta para el tuning de los modelos de este paso.

Hiperparámetros de estructura

- **Número de capas ocultas:** Corresponde a la cantidad de capas ocultas que conformarán la estructura de la red.
- **Número de unidades:** Indica la cantidad de neuronas que tendrá cada capa de la red. Se define como un vector cuya dimensión es la cantidad de capas ocultas y cada uno de sus elementos indica la cantidad de neuronas de la capa correspondiente.
- **Dropout:** Define el porcentaje de neuronas de cada capa que serán "apagadas" aleatoriamente durante cada "batch" de "back-propagation" para regularizar el modelo.

Hiperparámetros de control de aprendizaje

- **Tasa de aprendizaje (Learning rate):** Indica el tamaño de paso que se utilizará para controlar la actualización de los pesos de la red durante cada iteración de "back-propagation". Por definición suele ser uno de los hiperparámetros más importantes de un modelo de DL.
- **Tamaño del batch (batch size):** Corresponde a la cantidad de registros (filas) del dataset que se utilizarán para cada iteración de aprendizaje mediante "back-propagation".
- **Épocas (epochs):** Indica la cantidad de veces que se le mostrará el dataset de entrenamiento completo a la red durante el aprendizaje. Esto teniendo en cuenta que una época se define como una "pasada" completa sobre el dataset de entrenamiento luego de sucesivas iteraciones de "back-propagation".

Configuraciones adicionales

Debido a la complejidad asociada a la definición de un modelo de DL, no alcanza solo con definir los hiperparámetros del modelo sino que se deben tener en cuenta algunas consideraciones adicionales, las cuales se describen a continuación:

- La función de activación de todas las capas ocultas es la "Rectified Linear Unit" o ReLU, sin embargo, la función de activación de la capa de salida es "softmax" para poder realizar la clasificación multiclase.
- El optimizador utilizado para el entrenamiento de la red fue "Adam".
- La definición de las propiedades de la capa inicial de embedding es fija y se realiza en función del tamaño del vocabulario del dataset, es decir, la cantidad de tokens únicos, y asignando ciertos parámetros tales como la

dimensión deseada para los vectores de embedding y la longitud de las secuencias de texto que se ingresarán a la red, la cual suele asignarse acorde a la secuencia más larga presente en el dataset.

- Se incluyó el parámetro de "early stopping" para tener un control más fino sobre el proceso de entrenamiento y la capacidad de generalización de los modelos. Este parámetro tiene como función detener el entrenamiento cuando el desempeño de la red empieza a desmejorar y a acercarse más hacia el sobreajuste.

A continuación, se presenta la tabla que contiene los rangos de valores que se consideraron para cada uno de los hiperparámetros anteriormente mencionados:

Hiperparámetros Redes Densas de DL	
Hiperparámetro	**Valores**
N° Capas Ocultas	[3, 4, 5]
N° Unidades por capa	300 a 450
Dropout	[0.05, 0.1, 0.15, 0.2]
Learning Rate	10^{-4} a 10^{-1}
Batch Size	[1024, 2048, 4096, 8192]
Épochs	[10, 15, 20, 25, 30]

Tabla 11-3. Hiperparámetros redes densas de DL.

Similarmente al paso anterior, los rangos de la tabla 11-3 fueron determinados luego de realizar múltiples pruebas con diferentes espacios de exploración; en donde se encontró que este espacio búsqueda permitía obtener modelos con mejores desempeños con respecto a los demás espacios probados.

Para la implementación de este tipo de modelos en Spark se utilizó **Elephas**, el framework descrito en la sección 9.2.2 que permite correr redes de DL definidas con **Keras** sobre entornos de procesamiento distribuidos como Spark.

Entrenamiento, validación y evaluación de modelos

En este punto se presenta otra diferencia con respecto a los pasos anteriores, puesto que, en el proceso de obtención de redes de DL, cuando se utiliza el método de "early stopping" no es necesario realizar la validación de manera independiente como en el enfoque tradicional de obtención de modelos de ML. Esto debido a que el proceso de "early stopping" ya es en sí mismo un paso de validación, dado que para poder llevarlo a cabo se asigna una porción fija de los datos con la cual se realiza el control y seguimiento del desempeño del modelo durante el entrenamiento para detener el proceso cuando sea necesario. La

validación y selección de modelos se realiza utilizando dicha porción de los datos como punto de comparación.

Es necesario mencionar que, con respecto a las proporciones de los diferentes sets que se utilizan en el proceso, partiendo de los aproximadamente 9.5 millones de registros que tienen los datasets de cada idioma, se mantiene el esquema de 95% para entrenamiento de los modelos y 5% para validación, que en este caso correspondería al set de "early stopping".

El proceso de tuneo de las redes densas de DL para cada idioma se realizó siguiendo el enfoque de **búsqueda aleatoria**, el cual consiste en muestrear aleatoriamente conjuntos de hiperparámetros del espacio de búsqueda (tabla 11-3) hasta encontrar el que arroje el mejor modelo. Esto se definió de esta manera por una limitación computacional dado que el entrenamiento de este tipo de modelos tarda un tiempo considerable. Para este paso se utilizaron **15** evaluaciones sobre el espacio de búsqueda para la obtención del mejor modelo de cada idioma.

En cuanto al set de testeo y la métrica de desempeño se mantuvo el mismo enfoque que en los pasos N°3 y N°4 de la metodología, esto es, se utilizan los registros etiquetados como "reliable" para testear el mejor modelo obtenido de cada lenguaje y el desempeño en todo el proceso se mide utilizando el "weighted F1 score".

Guardado de resultados

De la misma manera que en los pasos anteriores, se guardan dos (2) archivos de texto (.csv) en el bucket de GCP, uno con las métricas de desempeño de la mejor red densa de cada idioma y otro con las predicciones de cada una de estas en el set de testeo.

Implementación

La ejecución de los experimentos de este paso se realizó utilizando un cluster de **nodo único** en Dataproc, esto debido a que cuando se pagan recursos de GCP con el crédito gratuito existen restricciones en la relación vcpus-memoria RAM que puede utilizarse. Y por la cantidad de memoria requerida para la obtención de estos modelos, no era posible completar la ejecución del código satisfactoriamente con una configuración como la utilizada en los pasos anteriores. La única manera que se encontró fue definir un cluster de nodo único con un número elevado de núcleos, lo cual permitió acceder a una mayor cantidad de memoria RAM en dicho nodo y se pudo realizar el entrenamiento sin problemas.

6. **Deep Learning – Redes Recurrentes:** En la sección anterior se mencionó que las redes recurrentes (RNNs) representan una topología avanzada de DL la cual está específicamente diseñada para abordar problemas en donde los elementos (features) de cada registro presentan una relación secuencial entre sí, tales como observaciones de una serie de tiempo o palabras en una secuencia de texto.

   Dado que la intención no es entrar en mayor detalle sobre la teoría y el funcionamiento de este tipo de modelos, se procede directamente a describir el procedimiento de implementación de los mismos como parte de la metodología propuesta.

   Consideraciones de diseño

   En línea con todos los pasos anteriores de modelado de la metodología, para este paso también se mantienen las mismas consideraciones de diseño que en estos.

   Datos de entrada

   Para este paso, al igual que en el paso N°5, también se utilizaron únicamente los datasets preprocesados que contienen los mono-gramas de los registros de cada idioma.

   Extracción de features

   De la misma manera que en el paso anterior, la extracción de features para la obtención de modelos de RNNs también se realizó mediante la utilización del método de "capa inicial de embedding", el cual ya fue descrito en detalle y no se hace necesario repetirlo en este punto.

   Definición de modelos de redes recurrentes

   Si bien se dijo que no se pretende explicar en detalle el funcionamiento de este tipo de modelos, es necesario mencionar que la versión "base" de las redes RNNs tiene un punto débil relacionado con la dificultad de mantener "memoria a largo plazo" cuando las secuencias a analizar son relativamente largas. Para entender esto se tiene que tener presente el modo en el que operan este tipo de redes, el cual se mencionó que ocurre analizando los elementos de cada secuencia (registro) de uno a la vez de izquierda a derecha a la vez que se considera el efecto de los elementos anteriores al actual mediante un mecanismo de memoria. Pues bien, cuando las secuencias se hacen largas, dicho mecanismo empieza a perder capacidad para retener el efecto de los elementos iniciales de estas, por lo que se dice que solo tiene memoria a corto plazo.

   Para mitigar lo anterior surgieron las redes LSTM y GRU, las cuales son redes recurrentes con modificaciones que les permiten contar con una capacidad de memoria aumentada y poder así procesar secuencias largas sin problemas.

Como consecuencia, estas variantes mejoradas terminaron desplazando a la versión base y son la opción por defecto a la cual se hace referencia cuando se habla de RNNs. Para el caso de la metodología propuesta, como parte de este paso se experimentó únicamente con redes LSTM.

Con respecto a los hiperparámetros y configuraciones consideradas para el tuneo de modelos de redes recurrentes LSTM, es necesario mencionar que estos fueron **prácticamente los mismos** que los utilizados en el paso anterior para las redes densas. Existiendo una única diferencia, la cual radica en la no utilización del hiperparámetro del "número de capas ocultas", esto debido a que la estructura estándar de una red recurrente suele definirse mediante solamente dos (2) capas, la capa recurrente y la capa de salida; por lo cual no se necesita de este hiperparámetro.

Por lo anterior, el espacio de búsqueda de hiperparámetros quedó definido según se muestra en la tabla 11-4.

Hiperparámetros Redes Recurrentes LSTM de DL	
Hiperparámetro	**Valores**
N° Unidades capa recurrente	300 a 450
Dropout	[0.05, 0.1, 0.15, 0.2]
Learning Rate	10^{-4} a 10^{-1}
Batch Size	[1024, 2048, 4096, 8192]
Epochs	[10, 15, 20, 25, 30]

Tabla 11-4. Hiperparámetros redes recurrentes LSTM.

Similarmente al paso anterior, la implementación de los modelos de redes recurrentes LSTM también se realizó mediante Elephas partiendo de una base definida en Keras.

Finalmente y a modo de comentario adicional, existe una variante conocida como "red recurrente bidireccional" la cual consiste en utilizar dos (2) capas recurrentes independientes, en donde la primera recorre la secuencia de izquierda a derecha y la segunda lo hace de derecha a izquierda, para acabar con la combinación de las salidas de ambas a fin obtener una única entrada a la capa de salida. Esto se realiza con la intención de considerar si es posible que el modelo obtenga información adicional al recorrer la misma secuencia en sentidos diferentes. Se realizaron experimentos con esta variante pero no se obtuvieron resultados que superaran significativamente a la LSTM unidireccional común, lo cual en conjunto con un incremento sustancial en el tiempo de entrenamiento llevó a descartar esta opción.

Entrenamiento, validación y evaluación de modelos

De la misma forma que las otras partes de este paso, el procedimiento de validación y evaluación de los modelos de redes LSTM también se realizó de igual manera que como fue descrito en el paso anterior, por lo que no hay nada para agregar.

Guardado de resultados

Para esta parte se mantuvo el mismo esquema que en todos los pasos de modelado que componen la metodología propuesta, esto es, se guardan el archivo con las métricas y el archivo con las predicciones de testeo en el bucket de GCP.

Implementación

La ejecución de este paso y todos sus experimentos asociados se realizó utilizando un cluster de Dataproc con las mismas especificaciones que las detalladas en el paso N°5.

12. Resultados y análisis

Habiendo descrito detalladamente todo el proceso de implementación de cada uno de los pasos que componen la metodología de solución propuesta, en esta sección se presentan y analizan los resultados obtenidos en cada uno de estos. Esto se hará siguiendo el mismo orden en el que se ubican dentro de la metodología.

Resultados análisis exploratorio de datos

Dado que en la sección 11 se listaron los pasos que se consideraron como parte del análisis exploratorio, a continuación se presentan los resultados obtenidos para cada uno de estos en conjunto con sus respectivos análisis y comentarios pertinentes.

- Con respecto a la exploración de las dimensiones del dataset general, en la figura 12-1 se muestran los resultados obtenidos:

```
El dataset tiene 20000000 registros y 4 columnas
```

Figura 12-1. Exploración de las dimensiones del dataset general.

Se puede observar que la cantidad de registros (filas) es de 20 millones, lo cual se alinea con todo lo mencionado hasta ahora a lo largo del documento.

- Seguidamente se presenta una muestra de las primeras cinco (5) filas del dataset, esto con el fin de dar una idea del aspecto general del mismo.

```
Muestra de los primeros 5 registros de la tabla:

                                               title label_quality
0  Hidrolavadora Lavor One 120 Bar 1700w  Bomba A...    unreliable
1                  Placa De Sonido - Behringer Umc22    unreliable
2              Maquina De Lavar Electrolux 12 Kilos     unreliable
3  Par Disco De Freio Diant Vent Gol 8v 08/ Frema...    unreliable
4  Flashes Led Pestañas Luminoso Falso Pestañas P...    unreliable

     language                  category
0     spanish  ELECTRIC_PRESSURE_WASHERS
1     spanish                SOUND_CARDS
2  portuguese           WASHING_MACHINES
3  portuguese        VEHICLE_BRAKE_DISCS
4     spanish             FALSE_EYELASHES
```

Figura 12-2. Muestra de algunas filas del dataset.

Es posible notar que el dataset se compone de cuatro (4) columnas, las cuales son:

  - **"title"**: que corresponde al título de cada publicación, el cual es el objeto principal de análisis del problema.

- **“label_quality”**: que índica si la categoría a la cual está asignada la publicación fue revisada y validada por el equipo del e-commerce.
- **“language”**: la cual índica el lenguaje en el cual está escrito el título de la publicación.
- **“category”**: esta contiene la categoría a la cual pertenece la publicación del registro en cuestión, es decir, esta es la variable objetivo (label) del problema.

Si bien solo se muestran unos cuantos registros, estos alcanzan para mostrar que el aspecto del dataset es coherente con todo lo mencionado hasta ahora en el documento. Se puede ver que hay registros de ambos idiomas, portugués y español, y que se tiene la columna que índica si la publicación fue validada, aunque en la muestra no se observen registros del tipo “reliable”.

- De la misma forma, también es posible referirse a la figura 12-2 para inferir la estructura de las columnas del dataset; en donde puede afirmarse que estas corresponden a campos del tipo **“string”** puesto que todas contienen información en formato de texto.
- Al dividir el dataset general en función de la columna “language” se obtienen los siguientes resultados:

```
Cantidad de registros por idioma:

      language     title  porcentaje
0  portuguese  10000000        50.0
1     spanish  10000000        50.0
```

Figura 12-3. Resultados al particionar por lenguaje.

En la figura 12-3 se observa que, en línea con lo que se mencionó en otras partes del documento, solo existen publicaciones en dos (2) lenguajes, portugués y español. Adicionalmente, es evidente que la partición según “language” es perfectamente simétrica, puesto que se obtienen dos datasets con 10 millones de registros cada uno.

- Si se realiza un análisis similar al anterior pero particionando por la columna “label_quality”, a nivel general se pueden obtener los siguientes resultados:

```
Cantidad de registros por calidad de la etiqueta:

   label_quality     title  porcentaje
0       reliable   1184245    5.921225
1     unreliable  18815755   94.078775
```

Figura 12-4. Resultados al particionar por “label_quality”.

De lo anterior es posible notar que solo aproximadamente el 6% de los registros del dataset cuentan con sus categorías validadas por el equipo del e-commerce. Debe tenerse presente que estos juegan un papel fundamental pues son los que se utilizan para conformar los conjuntos de testeo durante la experimentación. Sin embargo, debido a que todo el modelado se realizó de manera independiente para cada idioma, también se hace necesario inspeccionar la distribución de la columna "label_quality" en función de la columna "language". Los resultados obtenidos al hacerlo fueron:

Cantidad de registros según calidad de la etiqueta por idioma:

	language	label_quality	title	porcentaje
0	portuguese	reliable	693318	3.466590
1	portuguese	unreliable	9306682	46.533410
2	spanish	reliable	490927	2.454635
3	spanish	unreliable	9509073	47.545365

Figura 12-5. Resultados al particionar por "label_quality" y "language".

Se observa que para portugués se cuenta con el 3.5% de registros con categorías validadas, mientras que para español este valor se ubica alrededor de 2.5%. Si bien estos valores pueden parecer pequeños, la realidad es que para ambos casos la cantidad de registros validados se encuentra cerca de 500.000, lo cual constituye una muestra representativa del total de registros (aprox. 10M) de cada idioma y pueden utilizarse sin ningún inconveniente para el propósito de testeo que fueron definidos.

- Un dato no menor es que el dataset general cuenta con un total de **1588 categorías** (ver figura 12-6), esto da cuenta del nivel de complejidad del problema de clasificación multiclase que se está abordando. A manera de inspección se obtuvieron las categorías con mayor y menor cantidad de registros, las cuales se presentan a continuación:

El dataset tiene 1588 categorías

Cantidad de registros por categoría:

category	title	porcentaje
PANTS	35973	0.179865
COFFEE_MAKERS	35104	0.175520
BABY_CAR_SEATS	34163	0.170815
MUSICAL_KEYBOARDS	33222	0.166110
MATTRESSES	32967	0.164835

CONSTRUCTION_LIME_BAGS	206	0.001030
COLD_FOOD_AND_DRINK_VENDING_MACHINES	162	0.000810
PAINTBALL_SMOKE_GRENADES	154	0.000770
COMMERCIAL_POPCORN_MACHINES	141	0.000705
HAMBURGER_FORMERS	109	0.000545

Figura 12-6. Resultados 5 categorías con más y menos registros.

- Si bien se presentaron las cinco (5) categorías con mayor y menor cantidad de registros, también resulta interesante conocer el comportamiento de la distribución de las categorías del dataset; esto con el fin de conocer qué tanto difieren las categorías más comunes de aquellas con pocos registros. Los resultados obtenidos se muestran a continuación:

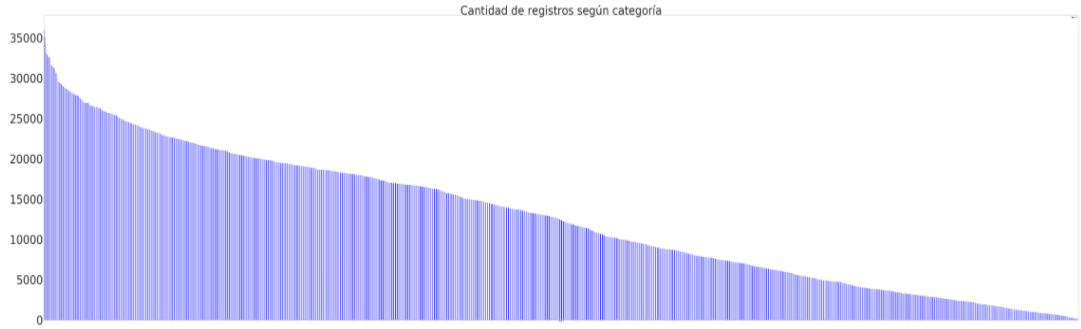

Figura 12-7. Distribución de las categorías del dataset.

En la figura 12-7 se observa que existe un decaimiento sostenido y constante de la cantidad de registros por categoría, lo cual permite descartar la idea de que existan unas cuantas categorías que acaparen la mayor cantidad de registros y dejen al resto con pocos.

De la misma forma que en incisos anteriores, dado que los modelos se obtienen de manera independiente para cada idioma, resulta también importante comparar el comportamiento de las distribuciones pero particionando de acuerdo a la variable “language”. Al hacerlo se obtuvieron los siguientes resultados:

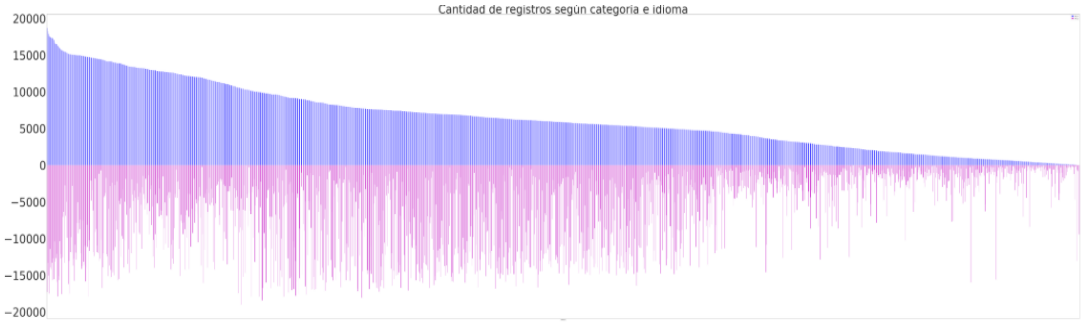

Figura 12-8. Distribuciones de las categorías según lenguaje.

Al observar la figura 12-8 resalta a la vista que el comportamiento de ambas distribuciones es totalmente diferente para cada idioma, lo cual refuerza el planteamiento de manejar cada lenguaje mediante modelos independientes que se enfoquen solamente en capturar las particularidades de un único idioma.

- Si bien se identificó que en el dataset general están presentes 1588 categorías, existe la posibilidad de que todas estas no se encuentren presentes en los registros de ambos idiomas; por lo que se inspeccionó si existen categorías que solo estén presentes en un idioma más no en el otro. Estos son los resultados obtenidos:

```
Hay 12 categorías que solo están en Español y son:

                    BATHROOM_FAUCET_SETS
                      CAR_EMERGENCY_KITS
                                CHARANGOS
                CONSTRUCTION_LIME_BAGS
                              GARDEN_SHEDS
                               GAS_BOILERS
                             HOCKEY_GLOVES
                                SAPO_GAMES
                                     SKIS
                          SNOWBOARD_BINDINGS
                        STORAGE_WATER_HEATERS
                     TOOL_ELECTRIC_SHARPENERS
Name: category, dtype: object
```

Figura 12-9. Categorías presentes únicamente en español.

```
Hay 14 categorías que solo están en Portugués y són:

                          CAR_AV_RECEIVERS
                         KITCHEN_FURNITURE
                CELLPHONE_REPAIR_TOOL_KITS
                               EROTIC_PUMPS
                PARTY_DECORATIVE_BACKDROPS
                           TRAILER_HITCHES
AUTOMOTIVE_SHOCK_ABSORBER_BUMP_STOPS
              AUTOMOTIVE_AIR_FILTER_BOXES
                                PAINTBALLS
                                FOOD_CARTS
                           ELECTRIC_BROOMS
                         PAINTBALL_O_RINGS
                               STADIOMETERS
                  PAINTBALL_SMOKE_GRENADES
Name: category, dtype: object
```

Figura 12-10. Categorías presentes únicamente en portugués.

Aunque en las figuras 12-9 y 12-10 se puede ver que efectivamente existen categorías presentes únicamente en uno de los idiomas, el número resulta no ser significativo como para tomar alguna acción con respecto a este hecho.

- Para finalizar el análisis exploratorio se incluyó una inspección rápida de la cantidad de palabras (tokens) que conforman los títulos de las publicaciones del dataset. A continuación, se presentan los resultados obtenidos:

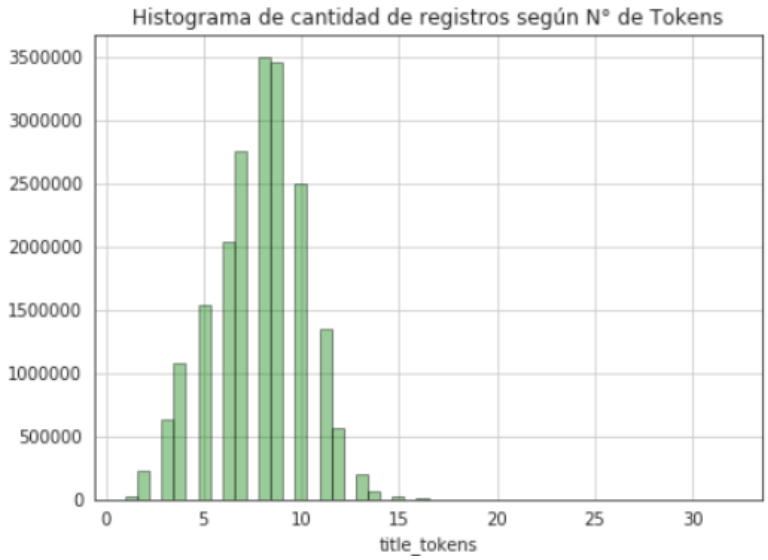

Figura 12-11. Histograma de registros según cantidad de tokens del título.

De la figura 12-11 se observa que, a nivel general, los títulos de las publicaciones se componen de alrededor de ocho (8) o nueve (9) palabras, esto sin realizar la tipificación según lenguaje.

Resultados modelos Naive-Bayes

Debe recordarse que para el caso de los modelos baseline utilizando el algoritmo de Naive Bayes se siguió un enfoque de experimentación que constaba de dos (2) maneras de extracción de features, razón por la cual se obtuvieron para cada idioma los mejores modelos en cada una de estas. Los resultados obtenidos luego de la experimentación y el tuneo de modelos se presentan a continuación:

Mejores modelos baseline con Naive Bayes				
Algoritmo	**Idioma**	**Extracción**	**F1 Training**	**F1 Test**
Naive Bayes	Español	Bolsa de palabras	0.824	0.857
Naive Bayes	Español	Hashing	0.885	0.853
Naive Bayes	Portugués	Bolsa de palabras	0.826	0.840
Naive Bayes	Portugués	Hashing	0.890	0.841

Tabla 12-1. Resultados mejores modelos usando Naive Bayes.

En la tabla 12-1 se observa que, en general, aunque se presentan los resultados correspondientes únicamente a los mejores modelos iniciales, estos logran ser suficientemente buenos tanto para training como para los datos del conjunto de testeo; lo cual representa un buen punto de partida para el objetivo de obtener una solución a la altura del desafío que supone la problemática de estudio.

En cuanto a la selección final del mejor modelo de cada idioma según el método de extracción de features, si bien en el conjunto de testeo no se logra apreciar una diferencia significativa, en el conjunto de entrenamiento si existe una diferencia a favor del método que utiliza "hashing" en ambos idiomas, razón por la cual estos fueron los elegidos tanto para español como para portugués como los modelos "vencedores" en este paso.

A continuación, se presentan los mejores hiperparámetros obtenidos luego del proceso de tuning para cada uno de los modelos vencedores de este paso.

Hiperparámetros modelos vencedores Naive Bayes			
Español		Portugués	
Hiperparámetro	Valor	Hiperparámetro	Valor
Smoothing	0.98	Smoothing	0.99

Tabla 12-2. Mejores hiperparámetros modelos vencedores.

Se puede observar que para ambos casos el valor resultante del hiperparámetro se encuentra cerca del límite superior, lo cual sugiere que quizás podría haber un valor por encima de 1 que permita llegar a un mejor resultado; sin embargo, por tratarse de los modelos baseline se decidió no continuar invirtiendo recursos adicionales en este paso.

Seguidamente se presenta un análisis de las predicciones para el conjunto de testeo realizadas por los modelos vencedores, de manera que sea posible estudiar el comportamiento de estos e identificar sus puntos fuertes y débiles para obtener "insights" que puedan ser útiles en futuras iteraciones.

- Se inicia presentando los valores de la métrica seleccionada, en este caso el F1-Score, para el conjunto de testeo de cada idioma y un agregado de esta a nivel general:

```
El f1-score en test para Portugués es: 0.8408470382208638
El f1-score en test para Español es: 0.8528423557590294
El f1-score en test general es: 0.8381839784230848
```

Figura 12-12. F1-Scores de los mejores modelos baseline.

Puede observarse que los valores presentados para cada idioma se corresponden con los mostrados inicialmente en la tabla 12-1. Adicionalmente, se obtiene que a nivel general la combinación de ambos modelos arroja un desempeño de 83.8% para la solución como un todo, un valor que no está nada mal teniendo en cuenta que son resultados obtenidos con los modelos iniciales.

- A modo de complemento se presenta la métrica de "accuracy", esto con el objetivo de tener un punto de comparación para contrastar los valores de la métrica utilizada para el tuneo y selección de los modelos:

```
El accuracy general en test para Portugués es: 0.815514670035972
El accuracy general en test para Español es: 0.8225683248222241
El accuracy general total es: 0.8184387521163273
```

Figura 12-13. Accuracy's de los mejores modelos baseline.

Se puede ver que para todos los casos el valor del "accuracy" resulta inferior al del F1-Score, lo cual deja entrever que existe oportunidad para mejorar los resultados y llegar así a una solución con mejor desempeño a nivel general.

- Además de ver el desempeño de los modelos vencedores mediante una métrica agregada, también resulta de utilidad explorar el comportamiento de la métrica a nivel categoría, esto con la finalidad de identificar casos particulares que pueden ser más difíciles que el resto y podrían abordarse de forma independiente. A continuación, se presentan los histogramas de la métrica de desempeño a nivel categoría para cada idioma:

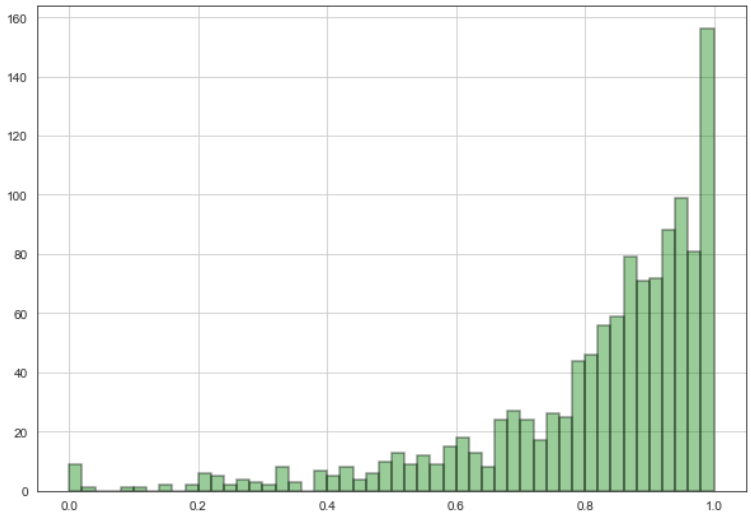

Figura 12-14. Histograma de f1-score a nivel categoría para portugués.

La figura 12-14 resulta bastante diciente sobre el comportamiento del modelo baseline (Naive Bayes) vencedor para portugués, pues en esta se observa que la

distribución de la métrica se orienta notablemente hacia la derecha, es decir, donde se ubican los mejores desempeños. Además, se evidencia que para la gran mayoría de categorías el desempeño está por encima de 80%, lo que se alinea con las métricas presentadas en las figuras 12-12 y 12-13 para este idioma.

De la misma forma, ahora se presenta el histograma de los desempeños del modelo vencedor para el idioma español:

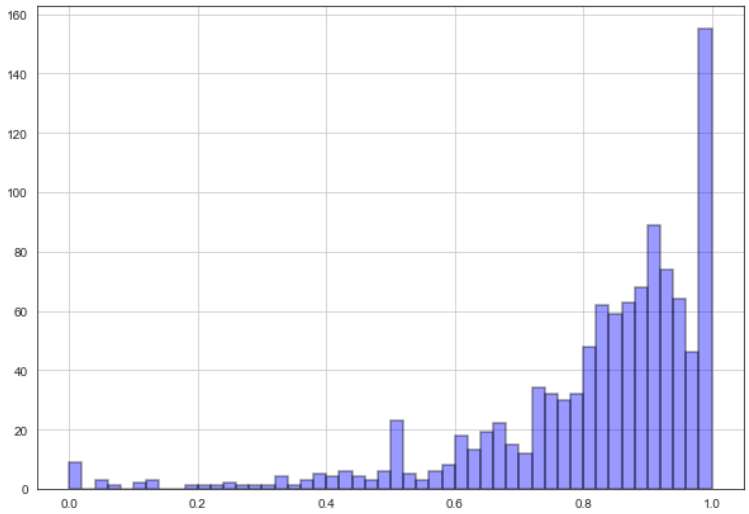

Figura 12-15. Histograma de f1-score a nivel categoría para español.

Tal y como se esperaba, la forma de la distribución resulta prácticamente igual a la mostrada para portugués, esto se explica debido a que los desempeños a nivel agregado (figuras 12-12 y 12-13) resultaron ser similares. Existe sin embargo un detalle a resaltar en este caso, y es que en los bines superiores a 90% del histograma se muestra un marcado descenso en la cantidad de categorías con dichos desempeños con respecto al caso de postugués.

- Finalmente, aunque los histogramas proveen un mejor nivel de detalle que las métricas agregadas, aún existe en estos una cierta agregación que tiene lugar en los bines que los conforman, por lo que podrían complementarse con un gráfico que muestre los resultados en el menor nivel de agregación posible, es decir, categoría a categoría. Es por esto que se presentan las figuras 12-16 y 12-17, en las cuales se muestran los desempeños para cada categoría por individual, lo cual da origen a una curva que resulta bastante interesante y evidencia a plenitud el comportamiento de los modelos de cada idioma.

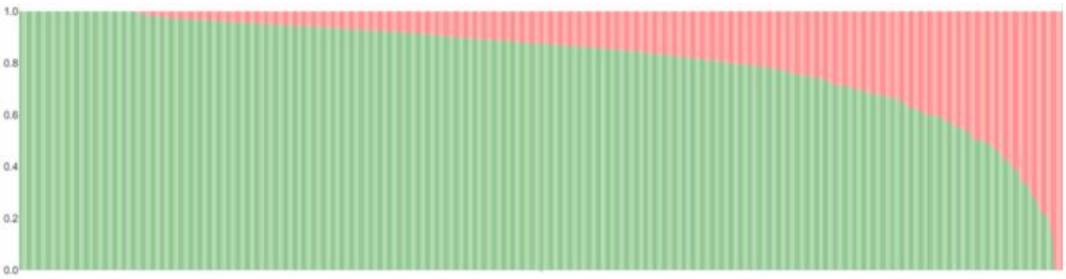

Figura 12-16. Curva desempeños categoría a categoría para portugués.

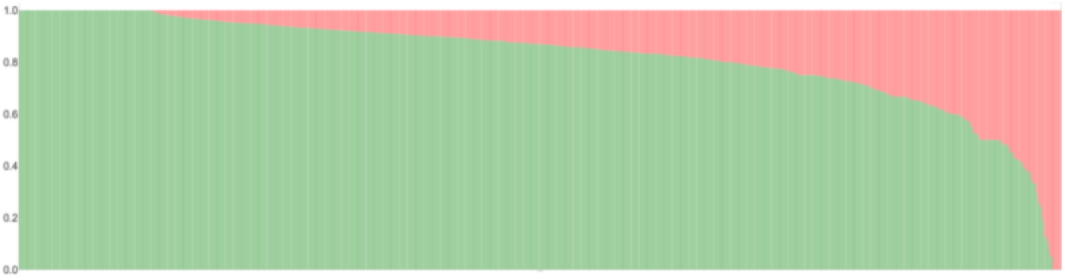

Figura 12-17. Curva desempeños categoría a categoría para español.

De ambas figuras resulta notable que existen una gran cantidad de categorías para las cuales los modelos tienen desempeño perfecto, luego estos empiezan a disminuir gradualmente hasta llegar a una muy pequeña cantidad de categorías para las cuales el desempeño es de 0%, estas pueden ser mayormente categorías con muy pocas publicaciones asociadas.

- A manera de comentario adicional, resulta interesante agregar que el proceso de tuning para obtener el mejor modelo baseline de un experimento, es decir, un idioma y un método de extracción de features, tardaba alrededor de 20 minutos. Por lo que para las cuatro (4) combinaciones posibles (2 idiomas y 2 métodos de extracción) resultaba de 1.5 horas aproximadamente.

Resultados modelos Random Forest

Tal y como se describió en la sección 11, el proceso de experimentación y obtención de modelos Random Forest siguió la misma mecánica que el de los modelos baseline, por lo que la presentación de resultados para este caso será bastante similar al caso anterior. A continuación, se presenta la tabla resumen con los desempeños de los mejores modelos obtenidos para cada experimento:

Mejores modelos con Random Forest				
Algoritmo	**Idioma**	**Extracción**	**F1 Training**	**F1 Test**
Random Forest	Español	Bolsa de palabras	0.424	0.497
Random Forest	Español	Hashing	0.461	0.569
Random Forest	Portugués	Bolsa de palabras	0.418	0.372
Random Forest	Portugués	Hashing	0.449	0.386

Tabla 12-3. Resultados mejores modelos usando Random Forest.

La tabla 12-3 presenta unos resultados que pueden parecer contradictorios a primera vista, esto debido a que resulta sorpresivo el hecho de que al utilizar un algoritmo con mayor poder predictivo y complejidad que el Naive Bayes como lo es Random Forest, los mejores desempeños obtenidos sean lo bastante pobres como para quedar rezagados con respecto a los obtenidos en los modelos baseline. La explicación a este fenómeno se dará más adelante cuando se presenten los hiperparámetros de los modelos "vencedores" de ambos idiomas.

Pasando a la selección de los modelos vencedores de cada idioma, en la tabla 12-3 se logra apreciar que existe una ligera ventaja en favor de los experimentos que utilizan el método de extracción basado en la técnica de "hashing", dicha ventaja logra apreciarse tanto para el conjunto de entrenamiento como para el de testeo; razón por la cual, similar al caso de los modelos baseline, estos experimentos son elegidos como los vencedores para este caso.

Los hiperparámetros que definen a los modelos vencedores se presentan a continuación:

Hiperparámetros modelos vencedores Random Forest			
Español		**Portugués**	
Hiperparámetro	**Valor**	**Hiperparámetro**	**Valor**
Max. Depth	30	Max. Depth	30
N. Estimators	350	N. Estimators	200
Max. Features	sqrt	Max. Features	sqrt
Max. Samples	0.8	Max. Samples	0.9
Min. Samples Leaf	2	Min. Samples Leaf	2

Tabla 12-4. Mejores hiperparámetros modelos vencedores.

Resulta interesante el hecho de que, para ambos idiomas, el mejor valor obtenido del hiperparámetro "maxDepth" es 30. Cuando se considera el hecho de que en ML solo se utilizan árboles binarios, este valor indica que los árboles construidos cuentan con 2^{30} nodos hoja; un valor que sobrepasa con creces la cantidad de registros que componen a los datasets de cada idioma. Esto sería un indicador de que, si bien los árboles de los

modelos Random Forest están notablemente sobredimensionados, aún así no resulta suficiente para obtener buenos desempeños y por tanto, la dificultad que suponen los datasets excede al poder predictivo de este tipo de modelos.

A continuación y aunque no resulta de mucho valor, se presenta el análisis exploratorio de las predicciones sobre el conjunto de testeo realizadas por los modelos vencedores.

- Iniciando con los valores de la métrica de desempeño para cada idioma y a nivel general, los cuales se corresponden con los presentados en la tabla 12-3:

```
El f1-score en test para Portugués es: 0.38625664135637683
El f1-score en test para Español es: 0.5692072264726461
El f1-score en test general es: 0.47153651760218035
```

Figura 12-18. F1-Scores de los mejores modelos Random Forest.

- Siguiendo con el "accuracy" para cada uno de los casos presentados en el inciso anterior:

```
El accuracy general en test para Portugués es: 0.37143129126894153
El accuracy general en test para Español es: 0.5245341975487191
El accuracy general total es: 0.4348998729148107
```

Figura 12-19. Accuracy's de los mejores modelos Random Forest.

- Ahora se mostrarán los histogramas con los resultados agrupados a nivel categoría para ambos idiomas:

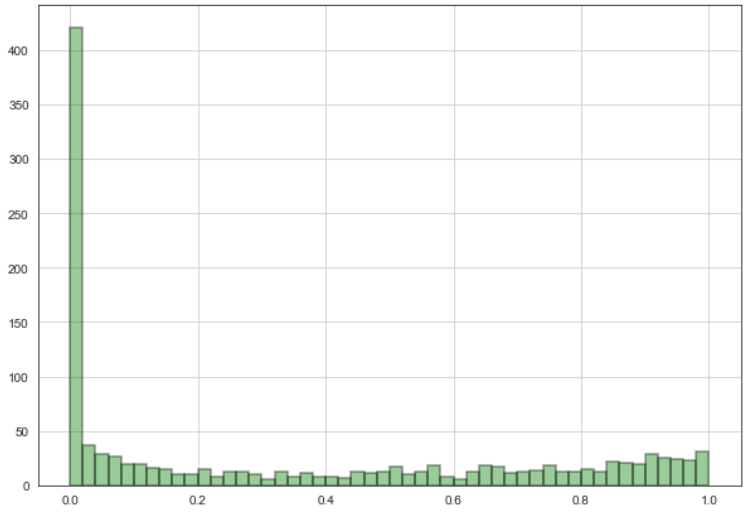

Figura 12-20. Histograma de f1-score a nivel categoría para portugués.

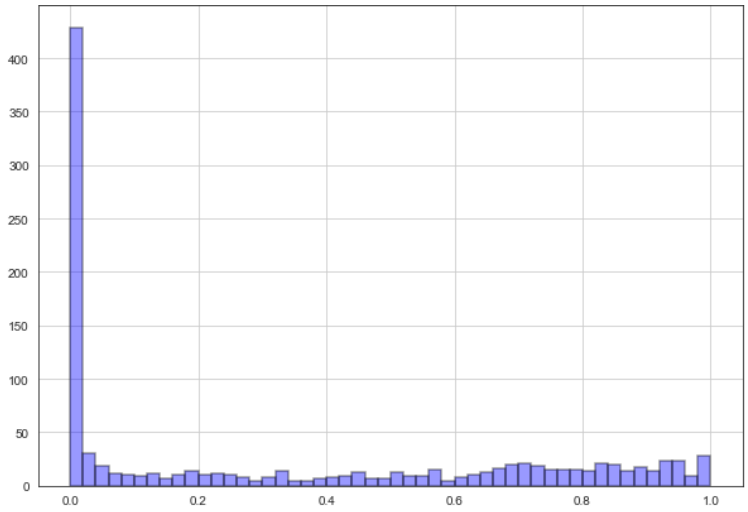

Figura 12-21. Histograma de f1-score a nivel categoría para español.

En las figuras 12-20 y 12-21 es posible observar que la diferencia entre el bin que representa las categorías con 0% de desempeño y el resto de bins de los histogramas es significativamente grande, lo cual reafirma el pobre desempeño obtenido de los modelos basados en Random Forest.

- Para finalizar se presentan las curvas con los desempeños categoría a categoría de ambos idiomas.

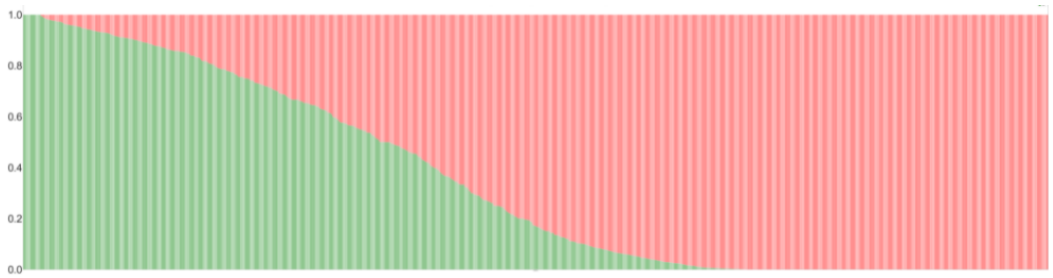

Figura 12-22. Curva desempeños categoría a categoría para portugués.

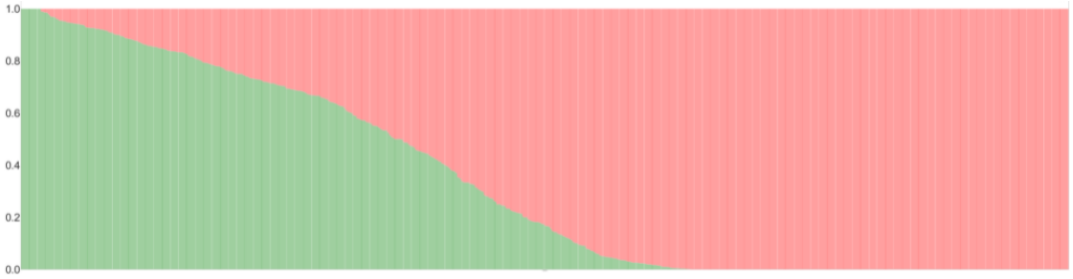

Figura 12-23. Curva desempeños categoría a categoría para español.

Si bien no hay mucho que comentar, los gráficos de las figuras 12-22 y 12-23 complementan a los histogramas presentados en el inciso anterior y dejan en evidencia que los modelos vencedores con Random Forest no performan bien para la gran mayoría de categorías del problema en ninguno de los idiomas.

Resultados modelos de redes densas de Deep Learning

En la sección 11 se describió la implementación de un enfoque de solución utilizando modelos de redes densas de DL, allí se planteó que el método de extracción de features sería únicamente mediante "word embeddings"; razón por la cual, para este caso solo se obtuvo un único mejor modelo para cada idioma luego del proceso de tuneo. A continuación, se presentan los resultados obtenidos al final de dicho proceso:

Mejores modelos con redes densas de DL				
Algoritmo	**Idioma**	**Extracción**	**F1 Training**	**F1 Test**
Dense DL ANN	Español	Word embeddings	0.904	0.893
Dense DL ANN	Portugués	Word embeddings	0.863	0.855

Tabla 12-5. Resultados mejores modelos con redes densas de DL.

La tabla 12-5 pone en evidencia unos resultados interesantes, puesto que son los mejores obtenidos hasta el momento de entre todas las familias de técnicas utilizadas. Debe considerarse que la real comparación se realiza contra los modelos baseline, puesto que los modelos de Random Forest quedaron bastante rezagados en términos de desempeño.

Para el idioma español, el mejor modelo de red densa de DL obtuvo desempeños de 90.4% en train y 89.3% en test, mientras que el modelo baseline vencedor obtuvo 88.5% y 85.3% respectivamente (ver tabla 12-1). Resulta evidente que el modelo de DL es superior para el idioma español ya que presenta mejores desempeños en ambos conjuntos, especialmente en testeo, donde la diferencia es de 4 puntos. De la misma forma, para el idioma portugués el mejor modelo de DL obtenido presenta desempeños de 86.3% en train y 85.5% en test, en cambio, el modelo baseline vencedor alcanza 89% y 84.1% respectivamente. En este caso es posible observar que la comparación resulta bastante ajustada y que el modelo de DL resulta vencedor solo en testeo.

Observando los resultados y tomando como referencia la comparación previa se pueden plantear los siguientes comentarios:

- Los modelos basados en redes densas de DL resultan superiores en cuanto a capacidad de generalización, esto puesto que para ambos idiomas alcanzan mejores desempeños en el set de testeo.

- Los modelos de DL son más consistentes que los modelos baseline, esto debido a que la diferencia entre sus desempeños de entrenamiento y testeo se mantiene pequeña en ambos idiomas. Efecto que no ocurre de la misma manera para los modelos baseline, los cuales presentan una caída de performance al pasar del set de entrenamiento al set de testeo.

Con respecto a los hiperparámetros que definen a los mejores modelos de DL obtenidos, estos se presentan en la siguiente tabla:

Hiperparámetros mejores modelos de redes densas de DL			
Español		Portugués	
Hiperparámetro	Valor	Hiperparámetro	Valor
N° Capas Ocultas	3	N° Capas Ocultas	4
N° Unidades por capa	Aleatorio	N° Unidades por capa	Aleatorio
Dropout	0.05	Dropout	0.05
Learning Rate	0.001	Learning Rate	0.0015
Batch Size	1024	Batch Size	1024
Epochs	25	Epochs	30

Tabla 12-6. Hiperparámetros mejores modelos de redes densas.

Un comentario que resulta pertinente agregar luego de observar la tabla 12-6 se relaciona con la designación "aleatoria" de la cantidad de neuronas en cada capa oculta de la red, esto se definió de esta manera para que cada capa pudiera tener un número diferente de unidades que el resto, permitiendo explorar así diferentes configuraciones.

Ahora se dará paso al análisis exploratorio de las predicciones de los mejores modelos de redes densas sobre el conjunto de testeo, el cual se presenta seguidamente:

- En correspondencia con las familias de modelos anteriores, se inicia con la exploración de la métrica de desempeño de interés, el F1 Score, tanto para cada idioma como a nivel general:

```
El f1-score en test para Portugués es: 0.8552303065548678
El f1-score en test para Español es: 0.8934120427620296
El f1-score en test general es: 0.8654443591551184
```

Figura 12-24. F1-Scores de los mejores modelos de redes densas de DL.

Lo más importante que puede extraerse de la figura 12-24 es el valor del F1 Score a nivel general, el cual resulta ser de 86.5% y por tanto, se ubica por encima del obtenido para la familia de modelos baseline (Naive Bayes), que es de 83.8% (ver figura 12-12). Lo cual confirma la superioridad en la capacidad de

generalización de los modelos de redes densas con respecto a las otras familias consideradas.

- Como siguiente paso se presentan los valores obtenidos en la métrica auxiliar, que es el "accuracy", la cual sirve como punto de comparación para evaluar los aciertos del modelo de la manera más general posible.

```
El accuracy general en test para Portugués es: 0.8346934595668942
El accuracy general en test para Español es: 0.878108150498954
El accuracy general total es: 0.8526909549966434
```

Figura 12-25. Accuracy's de los mejores modelos de redes densas de DL.

En la figura 12-25 se observa que el accuracy general de esta familia de modelos es de 85.2%, el cual resulta superior al obtenido para los modelos baseline, que es de 81.8% (ver figura 12-13). Adicionalmente, comparando los resultados a nivel general de los modelos de redes densas y lo modelos baseline para ambas métricas, es posible notar que la diferencia en ambos casos ronda los 3 puntos porcentuales a favor de los modelos de DL.

- La exploración continúa con el análisis de los histogramas en los que se presentan los resultados agrupados a nivel categoría, los cuales se muestran a continuación:

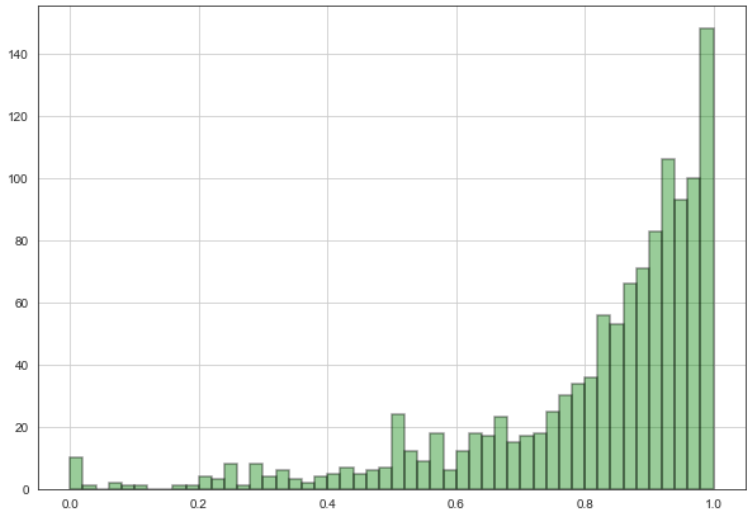

Figura 12-26. Histograma de f1-score a nivel categoría para portugués.

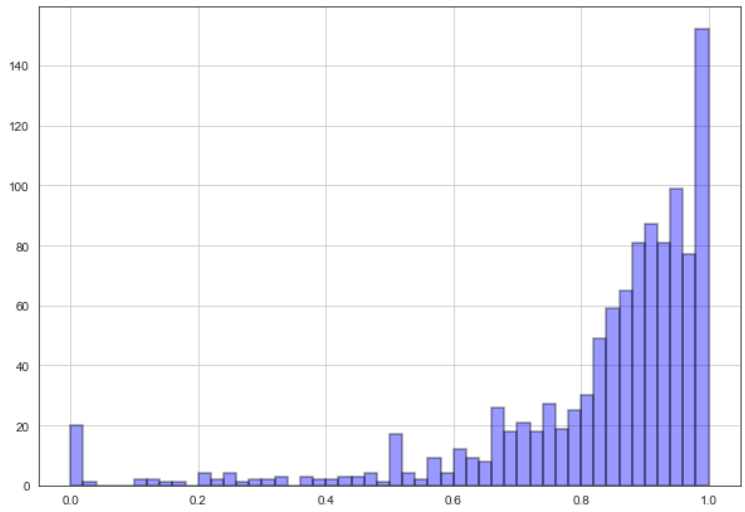

Figura 12-27. Histograma de f1-score a nivel categoría para español.

El comportamiento observado en las distribuciones de la métrica de desempeño a nivel categoría de los histogramas en las figuras 12-26 y 12-27 refleja el patrón esperado, el cual consiste en la presencia de los bines más altos, es decir, aquellos que representan la mayor cantidad de categorías, ubicados sobre los valores por encima de 70% de desempeño. Lo cual permite explicar con mayor detalle el origen de los resultados observados en las métricas agregadas de las figuras 12-24 y 12-25.

Un hecho a considerar es la existencia, para ambos idiomas, de un bin en solitario ubicado sobre el desempeño de 0%, lo que es indicador de que hay categorías para las cuales los modelos de redes densas no están acertando en ninguna de sus publicaciones.

- Para finalizar la exploración se presentan las curvas categoría a categoría, las cuales proveen el mayor nivel de detalle que tiene sentido considerar para evaluar el comportamiento de las predicciones de los modelos.

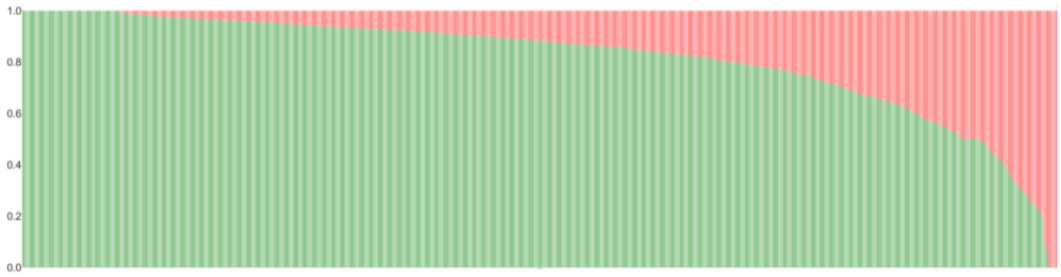

Figura 12-28. Curva desempeños categoría a categoría para portugués.

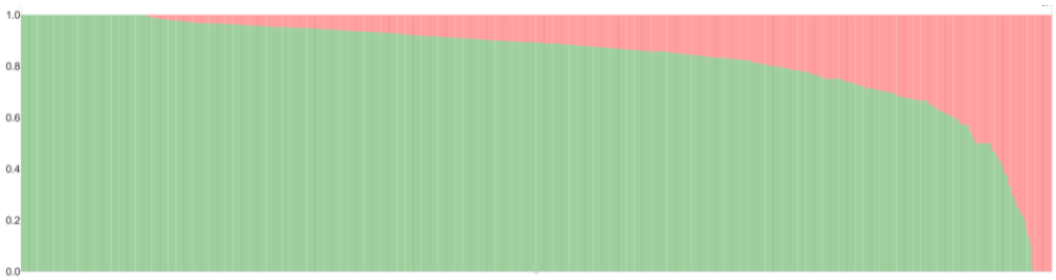

Figura 12-29. Curva desempeños categoría a categoría para español.

Aunque a este nivel resulta difícil realizar una comparación con respecto a las curvas de los modelos baseline, las figuras 12-28 y 12-29 refuerzan el planteamiento de los buenos desempeños obtenidos de los modelos de redes densas, puesto que para la mayoría de categorías se evidencian valores de la métrica por encima del 70% en ambos idiomas.

Resultados modelos de redes recurrentes de Deep Learning

Al intentar llevar a cabo el proceso de modelado siguiendo el enfoque basado en redes recurrentes de DL se presentó una situación que supuso un punto de inflexión para el cual se hizo necesario tomar una decisión, la situación es la que se describe a continuación.

En los modelos de redes densas, el proceso de entrenamiento para un conjunto de hiperparámetros, es decir, la evaluación de un punto del espacio solución, tomaba alrededor de 40 minutos; sin embargo, el entrenamiento de una única red recurrente tardaba aproximadamente 2.5 horas. Con lo cual, si se consideran las 15 evaluaciones propuestas para el tuneo en la sección anterior, el tiempo de ejecución del modelado asciende a 10 horas para el enfoque de redes densas y a 37 horas (1.5 días) para el de redes recurrentes.

Como consecuencia de lo anterior, la ejecución del proceso de tuneo para el enfoque de redes recurrentes implicaría un costo excesivamente alto de recursos de GCP dada la infraestructura y servicios utilizados para tal fin, el cual conllevaría a consumir todo el crédito restante disponible en la cuenta. Por lo cual, se decidió realizar una prueba utilizando dos (2) conjuntos de hiperparámetros muestreados al azar del espacio de búsqueda y analizar los resultados obtenidos.

Los resultados obtenidos de la prueba descrita anteriormente pusieron en evidencia que los modelos de redes recurrentes presentaban desempeños comparables a los que se obtuvieron con los modelos de redes densas, lo cual si bien deja la puerta abierta para obtener mejores desempeños utilizando redes recurrentes y siguiendo un proceso de tuneo, en este punto no es un camino que pueda ser transitado debido a la restricción de costo y crédito en GCP. Por lo tanto, el enfoque basado en redes recurrentes de DL

se descartó como opción viable para la implementación de la solución y no será considerado.

Selección del enfoque de modelado a utilizar

Habiendo cubierto el análisis de los resultados obtenidos luego de ejecutar los pasos de modelado para cada uno de los enfoques considerados (familias de modelos), se hace necesario realizar la selección del enfoque que se utilizará como parte de la solución propuesta. Es importante mencionar que para esto se tendrá en cuenta el desempeño de los modelos según la métrica elegida (F1-score), ya que la idea es elegir el tipo de modelos con mayor poder predictivo y capacidad de generalización para la problemática y datos en cuestión.

Procediendo con la selección, se inicia comentando que, como es de suponerse, los enfoques basados en Random Forest y redes recurrentes de DL están descartados; el primero por presentar bajo desempeño y el segundo por ser muy costoso a nivel computacional. Por tanto, la decisión final es entre los modelos basados en Naive Bayes (baseline) y los que utilizan redes densas de DL. Ante lo cual, debido a todo lo expuesto durante el análisis y comparación de resultados, resulta claro que los modelos de **redes densas** son los que cumplen de mejor manera con el criterio considerado y por tanto, conforman el enfoque seleccionado para dar solución a la problemática de análisis.

13. Conclusiones

Luego de haber realizado el proceso de definición y dimensionamiento del problema, diseño de la arquitectura, planteamiento y desarrollo de la metodología de solución, experimentación y modelado, análisis de resultados y finalmente, selección del mejor enfoque a utilizar; se hace necesario cerrar el presente proyecto con las conclusiones obtenidas al considerar los objetivos propuestos y los resultados obtenidos. Dichas conclusiones se presentan a continuación:

- Se pudo plantear y desarrollar una metodología sistemática cuya ejecución permitió obtener dos modelos de ML (1 por idioma) con niveles de desempeño lo suficientemente buenos como para ser considerados herramientas para la automatización del proceso de categorización de productos. Es posible concluir esto debido a que el F1-score combinado de los modelos del enfoque seleccionado es de 86.5%, lo cual en conjunto con los análisis realizados permitió evidenciar que estos presentan un desempeño adecuado en la gran mayoría de categorías; dejando solo unas pocas para ser revisadas por el equipo del e-commerce. Por lo cual, la carga de trabajo de dicho equipo se vería reducida notablemente.
- Se aplicaron satisfactoriamente algunas de las más conocidas estrategias de NLP para la extracción de "features" a partir de textos tales como, bolsa de palabras, TF-IDF y "word embeddings". Las cuales permitieron obtener, recibiendo como entrada texto libre, datasets estructurados que pudieron utilizarse como insumos de entrada para algoritmos de ML.
- Fue posible experimentar con diferentes técnicas de ML las cuales fueron utilizadas siguiendo un enfoque de complejidad incremental, es decir, iniciando con la de menos complejidad y luego ir probando otras cada vez más complejas y avanzadas.

  Durante este proceso se obtuvieron resultados ciertamente interesantes, como por ejemplo que los modelos basados en Naive Bayes, aunque se propusieron como los de menor complejidad (baseline), permitieron obtener muy buenos resultados ya que su funcionamiento subyacente se beneficia de como se generan los features en problemas de NLP. También resultó sorpresivo que los modelos basados en Random Forest, los cuales suelen tener desempeños generalmente buenos, se vieran notablemente afectados por los miles de "features" que conformaban los datasets de entrenamiento y terminaran arrojando resultados bastante pobres.

  Finalmente se experimentó con modelos de Deep Learning, los cuales pusieron en evidencia la razón por la que son el estándar para abordar problemas de NLP y permitieron obtener los mejores resultados de entre todos los algoritmos

considerados, hecho que llevo a que fueran seleccionados como el enfoque de modelado a utilizar como parte de la solución.

- Si bien el criterio definido para la selección del tipo de modelos a utilizar se basó en el mejor valor de la métrica de desempeño elegida, se pudo dar cuenta de que este no es el único factor que influye en la implementación de una solución de este calibre. Esto debido a que fue necesario descartar los modelos de redes recurrentes por su elevado tiempo de entrenamiento, lo que resultaba en un consumo excesivo de recursos de GCP. Razón que lleva a incluir la relación tiempo, costo y beneficio como una variable más a considerar al momento de proponer soluciones a problemas reales como el abordado en este proyecto.
- Se pudo diseñar y disponibilizar de manera exitosa una arquitectura de Big Data en GCP para soportar la solución desarrollada. La decisión de ir por el camino de Big Data se tomó teniendo en consideración el elevado volumen de datos del problema, que era de 20 millones de registros, la alta dimensionalidad asociada a problemas de NLP y la capacidad computacional requerida para entrenar modelos de ML utilizando tal cantidad de información.

  Adicionalmente, una arquitectura distribuida basada en la nube como la que se planteó permite que la solución propuesta sea robusta y escalable, lo que la hace estar a la altura de las exigencias y requerimientos de la industria para problemas de esta naturaleza.
- Aunque la solución final esté orientada a abordar el problema puntual de la categorización de productos en plataformas de e-commerce, la metodología propuesta resulta lo suficientemente general como para poderse emplear en otros problemas de NLP, especialmente de clasificación de textos. Esto se realizó de esta manera para dejar abierta la posibilidad de reutilizar el código en distintos escenarios a futuro.
- Como propuesta de trabajo a futuro se puede considerar la inclusión en la metodología de modelos de DL basados en “transformers”, los cuales para el momento de la entrega del presente documento son el estado del arte en problemas de NLP debido a sus excelentes desempeños, pero que durante el desarrollo del proyecto (2020) aún era difícil adaptarlos y usarlos en entornos distribuidos de Spark.

14. Referencias bibliográficas

Bird, Steven; Klein, Ewan; Loper, Edward. (2009). *Natural Language Processing with Python*.

Cevahir, A., & Murakami, K. (2016). Large-scale multi-class and hierarchical product categorization for an e-commerce giant. *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, 525–535.

Grosan, C., & Abraham, A. (2011). Machine Learning. In *Intelligent Systems Reference Library* (Vol. 17). https://doi.org/10.1007/978-3-642-21004-4_10

Harrington, P. (2012). *Machine Learning in Action (with Python)*.

Huang, S., Liu, X., Peng, X., & Niu, Z. (2012). Fine-grained product features extraction and categorization in reviews opinion mining. *Proceedings - 12th IEEE International Conference on Data Mining Workshops, ICDMW 2012*, 680–686. https://doi.org/10.1109/ICDMW.2012.53

K. Dalal, M., & A. Zaveri, M. (2011). Automatic Text Classification: A Technical Review. *International Journal of Computer Applications*, *28*(2), 37–40. https://doi.org/10.5120/3358-4633

Kozareva, Z. (2015). Everyone likes shopping! Multi-class product categorization for e-Commerce. *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, 1329–1333. https://doi.org/10.3115/v1/n15-1147

Kurtz, M. (2016). Introduction to E-commerce: Combining Business and Information Technology. In *BookBoon*. https://irp-cdn.multiscreensite.com/1c74f035/files/uploaded/introduction-to-e-commerce.pdf

Radulescu, C., Dinsoreanu, M., & Potolea, R. (2014). Identification of spam comments using natural language processing techniques. *Proceedings - 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014*, 29–35. https://doi.org/10.1109/ICCP.2014.6936976

Shen, D., Ruvini, J. D., Mukherjee, R., & Sundaresan, N. (2012). A study of smoothing algorithms for item categorization on e-commerce sites. *Neurocomputing*, *92*, 54–60. https://doi.org/10.1016/j.neucom.2011.08.035

Steinbach, M., Pi, V. I., Ku Mar, N., San, B., Newyork, F., Toronto, L., Tokyo, S., Madrid, S., Munich, M., Capetown, P., & Montreal, H. (2006). *Introduction to Data Mining*.

Gallardo, J. A. (2009). Metodología para la definición de requisitos en proyectos de data mining. Thesis (Doctoral), Facultad de Informática (UPM).

[Imagen representativa de la compra de productos en e-commerces]. Freepik. Recuperada de www.freepik.com.

[Imagen ilustración de la interfaz típica de publicaciones en e-commerces]. Mercado Libre. Recuperada de www.mercadolibre.com.

[Imagen ejemplo ilustrativo del proceso de categorización de productos en e-commerces]. Avalara. Recuperada de www.avalara.com.

[Imagen ejemplo ilustrativo de una aplicación de chatbot]. Freepik. Recuperada de www.freepik.com.

[Imagen ejemplo ilustrativo del proceso de aprendizaje supervisado]. Adobe. Recuperada de www. stock.adobe.com.

[Imagen ejemplo ilustrativo de un clasificador para detección de correos "spam"]. Medium. Recuperada de www.medium.com.

[Imagen proceso de la metodología CRISP-DM]. Medium. Recuperada de www.medium.com.