ESCUELA POLITÉNICA SUPERIOR DE MONDRAGON UNIBERTSITATEA
MONDRAGON UNIBERTSITATEKO GOI ESKOLA POLITEKNIKOA

Trabajo presentado para la obtención del título de
Titulua eskuratzeko lana

Máster universitario en Análisis de Datos, Ciberseguridad y Computación en la Nube

Datuen Analisia, Zibersegurtasuna eta Hodei-Konputazioa

Título del Trabajo *Lanaren izenburua*

DESARROLLOS DE INTELIGENCIA ARTIFICIAL PARA LA CLASIFICACIÓN Y FILTRADO AUTOMÁTICO DE TEXTOS DE TENTU Y SISTEMAS BACK-END ASOCIADOS

Autor *Egilea* ANDER BOLUMBURU CASADO

Curso *Ikasturtea* 2020/2021

Título del Trabajo *Lanaren izenburua*

DESARROLLOS DE INTELIGENCIA ARTIFICIAL PARA LA CLASIFICACIÓN Y FILTRADO AUTOMÁTICO DE TEXTOS EN TENTU Y SISTEMAS BACK-END ASOCIADOS.

Nombre y apellidos del autor
Egilearen izen-abizenak
BOLUMBURU CASADO, ANDER

Nombre y apellidos del/los director/es del trabajo
Zuzendariaren/zuzendarien izen-abizenak
AITOR OROBENGOA ORTUBAI
PEREZ, ALAIN

Lugar donde se realiza el trabajo
Lana egin deneko lekua
ISEA S.COOP

Curso académico
Ikasturtea
2020/2021

Resumen

El principal objetivo de proyecto ‘Desarrollos de inteligencia artificial para la clasificación y filtrado automático de textos en Tentu y sistemas Back-End asociados’ es el de investigar y desarrollar dos modelos que sean capaces tanto de clasificar temáticamente como de filtrar/censurar los contenidos recibidos por parte del cliente. En este caso, Tentu, la revista electrónica personalizada. Para lograr dicho objetivo se desarrollará un sistema totalmente autónomo desplegado en la nube que, sin ningún tipo de mantenimiento, será capaz de recopilar y preprocesar los datos con el fin de que sean entendibles por la red neuronal. Tras entrenar el modelo, este será ofrecido mediante un servicio REST desplegado de forma continua con el fin de que realicen sus predicciones. Durante esta etapa se realizará una importante toma de decisiones con el fin de proporcionar al cliente final un producto que cumpla con diferentes estándares tanto de calidad, como de seguridad y que aportarán valor al mismo.

Laburpena

‘Adimen artifizialaren garapenak eta haiekin lotutako back-end sistemak TENTUn testuak automatikoki sailkatzeko eta iragazteko asmoz gauzatuak’ proiektuaren xede nagusia sailkapen eta iragazketa automatikoan aurrerapenak egitea da, horretarako hori errazten duen adimen artifizial batzuen garapenean parte hartu da. Horretaz gain, modelo hauek bezeroei errazten dien bi zerbitzu garatu dira. Tentu aldizkari elektronikoa bere bezero nagusia izango da, horrela, bertan ikus daitekeen eduki guztia sailkatuta edo eta iragaztuta egongo da. Garapenean zehar, erabaki garrantzitsuak hartzeaz gain, funtzionalitate, test eta dokumentazio berrien garapena ere burutuko da, horrela aplikazioaren kalitatea handituz.

Abstract

The main objective of the project 'Developments of artificial intelligence for the classification and automatic filtering of texts in Tentu and associated Back-End systems' is to research and develop two models that are capable of both thematic classification and filtering/censoring the contents received from the client. In this case, Tentu, the personalized electronic magazine. To achieve this objective, a totally autonomous system will be developed and deployed in the cloud that, without any maintenance, will be able to collect and pre-process the data so that they are understandable by the neural network. After training the model, it will be offered through a REST service deployed continuously to make their predictions. During this stage, an important decision-making process will be carried out to provide the final customer with a product that complies with different standards of quality and security and that will provide value to the customer.

Tabla de contenido

Tabla de ilustraciones

Índice de tablas

1 Introducción

Este documento tiene como objetivo describir el proyecto realizado durante el Trabajo de Fin de Máster 'Desarrollos de inteligencia artificial para la clasificación y filtrado automático de textos en Tentu y sistemas Back-End asociados' en ISEA S. COOP. En el que se explicarán los siguientes apartados: Contexto del proyecto, sus objetivos, fases, desarrollo y finalmente los resultados obtenidos. Además, se incluye un pliego de condiciones en el que se describirán los elementos necesarios para la puesta en marcha del proyecto además de una memoria económica.

1.1 Contexto

El desarrollo de este proyecto se ha llevado a cabo en ISEA S.COOP, Centro de Innovación y Emprendimiento de carácter privado y sin ánimo de lucro. Especializado en el sector de los servicios empresariales, concretamente, en su departamento de desarrollo, junto con el equipo de desarrolladores.

Uno de los productos que ofrecen al público es Tentu, una revista electrónica personalizada que trata contenidos de fuentes RSS[1] de forma automática. Para clasificar los contenidos en diferentes categorías como pueden ser la economía o los deportes, actualmente hace uso de otro de los productos de ISEA, GAIA. Este un servicio de clasificación y filtrado de contenidos en la nube basada en los servicios que ofrece UZEI[2] para ello.

La empresa ya mencionada ISEA S. COOP, mediante el denominado proyecto 'Desarrollos de inteligencia artificial para la clasificación y filtrado automático de textos en Tentu y sistemas Back-End asociados', ha tratado de enfocar su rol en el desarrollo de dos modelos de inteligencia artificial que faciliten la labor de tanto la clasificación temática de textos como el filtrado/censura de estos. Cabe destacar que cada uno de estos modelos, van acompañados de un servicio REST desplegado de forma continua que ofrecen la predicción del modelo a los clientes.

Además, con el fin de añadir valor al producto, otro de los objetivos es desarrollar y desplegar en la nube una aplicación que sea capaz de recopilar diariamente textos correspondientes a diferentes temáticas para su posterior tratamiento y entrenamiento. De este modo, en caso de querer añadir una nueva categoría al modelo, solamente se deberán reunir diferentes fuentes RSS con el fin de que el recolector los capte.

Mediante el desarrollo de este sistema, se facilitará y agilizará de forma considerable la incorporación de nuevas temáticas a la revista ya que se automatizará por medio de inteligencia artificial la selección de las palabras más características de cada categoría. Evitando así el

[1] RSS es un formato XML para distribuir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.
[2] Enlace con más información sobre UZEI.

laborioso trabajo humano de seleccionar y ponderar cada una de las palabras correspondientes a cada temática.

Por otro lado, el hecho de que lleve varios años realizando las prácticas en la misma empresa, provoca que colabore en otros tipos de desarrollos. Estos pueden ser la propia revista Tentu, en este caso en la adaptación y publicación de la aplicación Flutter[3] al sistema operativo iOS o el producto Evidence Box[4], en el que he tenido que trabajar en el cambio de proveedor cloud para el back-end.

Para llevar a cabo dicho proyecto, ISEA cuenta con un equipo de ingenieros especializados en el desarrollo de tareas que conciernen al mismo, que se realizará en el departamento de desarrollo.

En cuanto al código, este es gestionado mediante un sistema de control de versiones. El objetivo es implementar diferentes testeos, que se realicen automáticamente en cada actualización de código fuente, de modo que se compruebe la robustez del avance que aporta dicha actualización de código.

Otro de los objetivos es mantener informado al equipo de desarrollo en caso de error, ya que, si algún tipo de testeo fallase al ser ejecutado, lo ideal sería, que mandase una notificación de dicho error a las cuentas de correo electrónico de todo el equipo, de modo que se cumpliría el objetivo de mantener a todos informados a cerca de la calidad del código.

En caso de que el aporte de código pase los testeos, sería adecuado que mediante herramientas de integración continúa como Gitlab CI/CD[5] se desplegase automáticamente en el servidor una versión con el nuevo aporte de código.

Por último, con el fin de llevar una adecuada gestión de tareas, se están utilizando dos herramientas. Por un lado, se ha realizado al comienzo del proyecto un diagrama de Gantt general con las tareas globales, para realizarlo se ha empleado la aplicación web *Team Gantt*[6]. Por otro lado, la gestión que conciernen a cada tarea global se ha realizado mediante un sistema de tablas que contiene la herramienta de control de versiones.

1.2 Antecedentes

Teniendo en cuenta la breve contextualización realizada anteriormente, a lo largo de este apartado se introducirán los elementos básicos los cuales son clave para el inicio del proyecto.

[3] Enlace con más información sobre Flutter.
[4] Enlace con más información sobre Evidence Box.
[5] Enlace con más información sobre Gitlab CI/CD.
[6] Termino que se refiere a la aplicación web diseñada para mejorar la comunicación y colaboración dentro de un equipo.

1.2.1 ISEA S. COOP.

Como se ha mencionado en el apartado anterior, Innovación en Servicios Empresariales Avanzados – ISEA S.COOP. es un centro de innovación y emprendimiento de carácter privado y sin ánimo de lucro especializado en el sector de los servicios empresariales, promovido por la división de ingeniería y servicios empresariales de la Corporación Mondragón.

ISEA S. COOP. es un agente científico tecnológico integrado en la red vasca de ciencia, tecnología e innovación. En la actualidad forma parte de la Agencia Vasca de Innovación. Adicionalmente, ISEA es un agente homologado del Servicio Vasco de Emprendimiento de SPRI (Sociedad para la Transformación Competitiva), dependiente del Gobierno Vasco.

Se trata de una entidad declarada de utilidad pública por la Consejería de Justicia, Empleo y Seguridad Social del Gobierno Vasco.

En cuanto a la misión de ISEA S. COOP. se refiere, esta radica en mejorar la competitividad del sector de los servicios empresariales mediante la potenciación del desarrollo tecnológico, la innovación y el emprendimiento de nuevas actividades empresariales.

Una vez introducida tanto la empresa, como su misión, en el siguiente apartado se procederá a explicar tanto la necesidad del desarrollo del producto, como el modo en el que se abastecerá esa necesidad.

1.3 Problemática

En el siguiente apartado de procederá a explicar el problema general que se va a tratar de solucionar mediante el desarrollo del producto.

1.3.1 Problema principal

Más del 80% de la información digital disponible en internet es información no estructurada en forma de textos y documentos [1]. En la economía del conocimiento, la reutilización de la información del sector público y privado presenta un considerable potencial económico pues las empresas infomediarias (535 empresas en España) analizan y tratan información del sector público y/o privado para crear productos de valor añadido (1.550 M de € de facturación) destinados a terceras empresas o a la ciudadanía.

El proceso de reaprovechamiento de tal información por las empresas del Sector Infomediario (Generadores de Información, Agregadores de Información, Capacitadores, Enriquecedores) precisa del desarrollo de una serie de hitos: analizar el formato, preparar y ordenar la información a través de la catalogación y la categorización.

Esta problemática es singularmente relevante en el aprovechamiento de la información abierta, pues es preciso resaltar que existe un importante volumen de información Open Data

[2] que no consigue emerger hacia la sociedad pues su contenido es desconocido para los ciudadanos y las empresas.

Por otro lado, el régimen de actuación legal de la prensa on-line, las redes sociales o cualquier otro recurso digital que ofrece puerta abierta a la participación de las personas está determinado por el corpus europeo y español en materia de telecomunicaciones, comercio electrónico, Sociedad de la Información y eAdministración.

Para que la participación en dichos medios sea acorde con el conjunto de dicho corpus legal, se debe garantizar el filtrado de contenidos para que los mismos no incorporen elementos atentatorios a la imagen y privacidad de las personas o incorporen actividades ilegales (promoción de la prostitución, venta de drogas, apologías diversas, etc.). La cumplimentación manual de las labores de filtrado será poco factible y, en todo caso, costosa, por lo que se estima necesaria la incorporación de tecnologías de filtrado de contenidos basados en procesamiento de lenguajes naturales y recursos lexicográficos especializados que posibiliten la ejecución de labores de filtrado automático o supervisado.

En una sociedad donde las tendencias son tan volátiles, resulta interesante ser capaces de clasificar contenidos en base a esa tendencia de forma rápida. Es ahí donde se presenta una de las debilidades del sistema GAIA. Actualmente, para su clasificación temática, debe pasar por un equipo humano que selecciona y pondera las diferentes palabras correspondientes a la nueva categoría.

El principal objetivo de este proyecto es la reducción de la mano de obra humana en trabajos repetitivos y con poco valor añadido. Dejando en manos del software la recolección de textos y la selección de palabras en las de la inteligencia artificial. Además, mediante la solución planteada, GAIA sería mucho más rápida ofreciendo esas nuevas temáticas, frente a los 4 o 5 meses que tarda actualmente con el sistema actual desarrollado por UZEI.

Una vez presentado el problema a resolver, en el siguiente apartado se explicará el producto con el que se intentará abastecer esa necesidad.

1.3.2 Producto

El producto en el que se ha pensado con el fin de poder abastecer esa necesidad es la ya existente GAIA, que consiste en unos servicios web soportados en tecnologías de clasificación y filtrado automática de textos proporcionados por UZEI (Centro Vasco de Terminología y Lexicografía) los cuales clasificarán y/o filtrarán los textos recibidos por la persona usuaria. Dichos servicios serán ofertados tanto en euskera como en castellano.

Además, ofrece la clasificación del texto en diferentes estándares como Eurovoc, CDU y NewsCodes, la decisión de la elección de estos estándares será explicada más adelante en el apartado 1.3.2.2.

En la siguiente ilustración, se pueden visualizar los actores participantes en el módulo de interoperabilidad.

Ilustración 1: Actores principales de GAIA

A modo de resumen, se podría decir que el cliente manda al servicio GAIA el texto que desea clasificar o filtrar, este se comunica con UZEI, del que espera una respuesta con el texto clasificado y/o filtrado y finalmente hace eco de esa respuesta al cliente.

Como se ha especificado en el apartado anterior, el principal problema a resolver por el producto es la evitación de la labor humana a la hora de seleccionar y ponderar las palabras correspondientes a cada temática. Es por ello por lo que el objetivo de este proyecto es el de diseñar y desarrollar un sistema que sustituya o complemente el servicio ofrecido por UZEI.

Tras presentar el producto en cuestión, en las siguientes líneas se analizará el objetivo geográfico y temático que se pretende alcanzar mediante el producto.

1.3.2.1 Alcance geográfico y temático

En cuanto a su alcance, debido a que GAIA soporta hoy en día el euskera y castellano se podría decir que su alcance geográfico se limita a territorio nacional. Incluso cabría la posibilidad de ofertar el servicio en Latinoamérica, aunque conllevaría un ajuste de los clasificadores/filtradores ya que las palabras malsonantes, por ejemplo, no son las mismas. Por otro lado, en caso de que en un futuro se aplicase el soporte en inglés, se podría expandir a otros países.

En cuanto al alcance temático, GAIA cuenta con tecnologías de clasificación automática de textos, también denominada categorización de textos o *topic spotting* [3], que opera sobre la base de un motor lematizador[7] inteligente y taxonomías[8] temáticas desarrolladas por UZEI-Centro Vasco de Lexicografía. De esta manera, es capaz de asignar automáticamente un destino temático especifico a los contenidos recibidos por parte del cliente según 49 temáticas, las cuales se irán ampliando con el paso del tiempo llegando a un máximo de 151.

Una vez explicado el alcance tanto geográfico como temático, se analizará la estructuración que se ha decidido llevar a cabo para las temáticas tratadas por GAIA.

[7] Término que se refiere la lematización, un proceso lingüístico que consiste en, dada una forma flexionada, hallar el lema correspondiente, es decir, el núcleo de esa palabra.
[8] Término que se refiere en su sentido más general, a la ciencia de la clasificación.

1.3.2.2 Estructuración temática de GAIA

Con el fin de tener una estructuración temática correcta, se ha procedido a analizar las expectativas de los potenciales clientes, expertos, comerciales y técnicos del Consorcio que han planteado la necesidad de llevar a cabo el desarrollo del servicio GAIA. Estas expectativas han concluido con el desarrollo de una primera lista de temáticas en función de las necesidades de estos.

Tras realizar dicho listado, se han analizado un conjunto de Iniciativas Europeas en materia de normalización y lexicografía; entre los que destacan:

- La Clasificación Decimal Universal o CDU [4].
- SKOS [5] (siglas de *Simple Knowledge Organization System*) es una iniciativa del W3C en forma de aplicación de RDF.
- Eurovoc [6], tesauro multilingüe y multidisciplinario que abarca la terminología de los ámbitos de actividad de la UE.
- Los resultados de las soluciones de las iniciativas de la Unión Europea en materia de interoperabilidad para las administraciones públicas europeas (ISA).
- El sistema de clasificación de la UNESCO.
- El sistema de clasificación de NEWSCODES de IPTC [7]: IPTC es un consorcio internacional constituido por las agencias de prensa y las empresas de comunicación más importantes del mundo. Entre ellas, se pueden referir:
    - *Alliance Européenne des Agences de Presse.*
    - ANPA (ahora NAA).
    - FIEJ (ahora WAN).
    - *North American News Agencies* (*Associated Press, Canadian Press and United Press International*).

Una vez analizadas las diferentes opciones se ha llegado a las siguientes conclusiones:

- Selección de los estándares más oportunos para el proyecto:
    - El sistema de clasificación de NEWSCODES de IPTC.
    - El sistema de clasificación Eurovoc, tesauro multilingüe y multidisciplinario que abarca la terminología de los ámbitos de actividad de la UE.
    - La Clasificación Decimal Universal o CDU.
- Se ha realizado una lista de 215 temáticas seleccionadas y se ofrecen sus correspondencias en los diversos estándares seleccionados.
- Finalmente, se ha realizado un listado de temáticas prioritarias para el desarrollo del proyecto GAIA.

Tras dejar fijada la estructuración temática a seguir, es importante analizar los orígenes potenciales de los contenidos a clasificar y/o filtrar.

1.3.2.3 Orígenes potenciales de los contenidos a clasificar/filtrar

Como ha sido mencionado, se prevé que gran parte de los contenidos a clasificar/filtrar por los servicios provengan de empresas infomediarias, los cuales utilizarán el contenido categorizado para dar un valor añadido a los servicios.

Estas organizaciones pueden utilizar datos de los siguientes orígenes:

- Redes RSS.
- Contribuciones de Instituciones y Empresas.
- Contribuciones de Asociaciones, ONG-s y demás entes vinculados al voluntariado.
- Contribuciones ciudadanas.

De la misma manera, es preciso enunciar que existe un importante volumen de información que no consigue emerger hacia la sociedad o que encuentra dificultades para ello.

Un buen ejemplo de cliente potencial es Tentu – La revista electrónica personalizada, la cual capta a través de más 700 fuentes RSS diferentes contenidos para tematizarlos y posteriormente destinarlos a colectivos interesados. Por tanto, podría considerarse un mecanismo de emparejamiento de la información aportada por un "colectivo interesado en la creación de la información" y entregada a un "colectivo interesado en su recepción".

1.3.2.4 Facilidad de uso y accesibilidad de GAIA

El servicio GAIA contiene varios componentes a disposición de los clientes:

Por un lado, la aplicación web con la que los usuarios reciben información sobre ciertos aspectos del servicio. Estos pueden ser sus utilidades, tarifas de pago o incluso una zona de demostración con el fin de comprobar que el funcionamiento del servicio se adapta a sus necesidades.

En caso de estar interesados en la contratación, la misma aplicación web sirve para realizarla además de gestionar el uso de los servicios, dando la posibilidad de limitar el número de peticiones por minuto con el fin de no sobrepasar la facturación establecida.

Por otro lado, los servicios web los cuales están documentados mediante Swagger. Esta aplicación permite al cliente, en este caso un desarrollador, conocer diferentes aspectos de la API como podrían ser:

- Parámetros que debe enviar para recibir la respuesta correspondiente.
- Parámetros que recibirá el cliente en la respuesta.
- Proporciona al desarrollador un pequeño *sandbox*[9] en el que podrá realizar pruebas.
- Códigos de error para conocer en todo momento los que se puedan generar.

[9] Termino que se refiere a un mecanismo para ejecutar programas con seguridad y de manera separada.

Dicha documentación está disponible a través siguiente enlace.

Para comprender la relevancia de una buena documentación se recomienda la lectura del artículo citado en la bibliografía [8].

Como se ha pensado en el desarrollador, también se ha publicado una librería NPM[10]. de modo que, para hacer uso de los servicios, el usuario solo tendrá que contratarlos e introducir la API Key [11] a la hora de importar la librería mediante la herramienta de gestión de dependencias.

Una vez hecho esto, será posible realizar las consultas y recibir las respuestas en apenas una línea de código. Esto facilita mucho las cosas, ya que el desarrollador no tiene por qué preocuparse de implementar otra librería externa para realizar las peticiones al servidor. Además de reducir drásticamente la cantidad de líneas de código de la aplicación, algo a agradecer tanto por parte del desarrollador actual, como el que pueda incorporarse en un futuro ya que esto facilitará de forma considerable su comprensión.

El enlace hacia la dependencia publicada en NPM está disponible aquí.

1.3.2.5 Innovación

En cuanto a la innovación, que en cualquier caso es importante, resulta conveniente señalizar los principales aspectos a través de los cuales el presente proyecto refleja innovación.

Por un lado, GAIA ofrece una vía novedosa de aprovechamiento masivo de información pública y privada. Lo cual como se ha comentado anteriormente beneficia a las compañías infomediarias, las cuales, mediante la herramienta ofrecerán datos más precisos a sus clientes.

Por otro lado, cabe destacar que, el proyecto, se alinea con las indicaciones de la unión europea en su comunicación 2013/37/EU [9] referente a las *"oportunidades de creación de valor en el open data están en el desarrollo de aplicaciones que combinen datos públicos con datos privados y sean empaquetados para satisfacer necesidades específicas de segmentos de clientes".*

Tras este análisis a cerca de la innovación que aporta la realización del proyecto, se procederá a explicar el estado actual en el que se encuentra el producto.

[10] Término para referirse a una herramienta de gestión de dependencias.

[11] Una API Key es un identificador que sirve como medio de autenticación de un usuario para el uso de los servicios proporcionados por GAIA.

1.3.3 Estado inicial

A la hora de comenzar el desarrollo de este proyecto, la aplicación web y los servicios ya están desarrollados al completo. Además, un pequeño prototipo del modelo de inteligencia artificial ha sido realizado con apenas 8 categorías.

Es por ello por lo que el desarrollo de este proyecto se centra parcialmente en el desarrollo del software que posibilite la recolección de diferentes fuentes RSS de forma automática. El desarrollo de dicho software conlleva una importante toma de decisiones como por ejemplo donde será alojado tanto el software como los datos recopilados o en qué condiciones o periodicidad será accionado.

Por otro lado, se debe investigar sobre diferentes algoritmos con el fin de obtener el modelo que mejor resultado de con los datos recopilados. Tras conocer el algoritmo que mejor se adecúa a los datos, se analizarán opciones con las que poder ofrecer dichos modelos a los usuarios.

Una vez analizado el estado actual del producto, es hora de analizar los retos que supone el hecho de realizarlo.

1.3.4 Retos

Tras analizar el estado inicial del proyecto, se procederá a definir los retos que propone el desarrollo.

Atendiendo a lo establecido tanto en la introducción como en la contextualización de este informe, el principal reto que presenta el proyecto consiste en la realización del software que se encargue de recopilar los artículos de las diferentes fuentes RSS.

Para ello un primer reto ha consistido en realizar un exhaustivo análisis de las tecnologías y software disponibles y, a su vez, alineados con las necesidades y objetivos empresariales. Además, en base a ese análisis se tomarán unas decisiones que afectarán al transcurso del proyecto.

Por otro lado, otro de los retos consiste en el desarrollo de diferentes modelos y su comparativa para identificar el que mejor se comporte con los datos recopilados por el software.

Por último, se debe asegurar la integración de la aplicación y los servicios. En otras palabras, se debe lograr la correcta comunicación entre ambos elementos de manera a posibilitar una explotación racional del conjunto.

1.4 Objetivos

Tras plantear los retos que presenta el proyecto, en el siguiente apartado se definirán los objetivos que se pretenden lograr a través del proyecto.

En primer lugar, y como objetivo principal resulta imprescindible cumplir las competencias establecidas tanto por la universidad como la empresa, ya que el cumplimiento de dichas competencias derivará en un buen desarrollo del proyecto y, por lo tanto, del producto.

En segundo lugar, es necesario un análisis y desarrollo del software que posibilite la recopilación de los artículos, así como de los recursos disponibles en la empresa. Para ello se hará un detallado proceso de selección, para así garantizar el funcionamiento correcto de la aplicación.

Por otro lado, se debe investigar acerca de que algoritmo es el que mejor se adecúa a los datos obtenidos por la aplicación desarrollada.

Volviendo a los retos anteriormente mencionados, una parte del proyecto se basa en el desarrollo de un software capaz de recopilar artículos de diferentes fuentes RSS de forma automática. No obstante, resultaría interesante que dicha aplicación fuese desplegada en la nube. De esta forma puede ser accionada automáticamente mediante un servicio externo como Google Cloud Scheduler[12] y el equipo de desarrollo no debe estar pendiente de la recolección de datos.

Asimismo, otro de los objetivos del presente proyecto, se refiere al desarrollo de dos modelos de inteligencia artificial (uno para la clasificación y otro para el filtrado). Para ello se procederá a analizar diferentes algoritmos con el fin de obtener el modelo que mejor resultado proporcione.

Una vez se tienen los modelos, resulta adecuado indicar que se estudiará la mejor forma de proporcionar al cliente la posibilidad de hacer uso de dichos modelos.

Desde un punto de vista objetivo, es conveniente admitir que la existencia de controles de calidad cada vez que se actualiza el código fuente, garantiza la estabilidad y robustez de la aplicación. Es por ello por lo que el cuarto objetivo consiste en realizar los testeos de cada funcionalidad desarrollada, de modo que se garantice dicha robustez.

Es muy importante realizar un buen comentario/documentación en el código, ya que el futuro desarrollador que se incorpore al proyecto debe de ser capaz de entender el código para llevar a cabo sus labores. Es por ello por lo que otro de los objetivos consiste en realizar una buena documentación del código.

Además de la documentación del código, resulta conveniente documentar cada punto de entrada ofrecido mediante los servicios, de modo que el usuario sepa en todo momento los parámetros de entrada y salida de cada target[13].

En cuanto al ámbito académico del proyecto, sería interesante analizar y formarse sobre herramientas desconocidas, siempre que sean útiles para el desarrollo, con el fin de aportar valor tanto al producto final como al proyecto en sí mismo.

[12] Enlace con más información sobre Google Cloud Scheduler.
[13] Término utilizado para referirse a cada una de las rutas disponibles en los servicios web.

Para finalizar, se debe de intentar en la medida de lo posible llevar un orden con las tareas asignadas y hacer uso de diferentes herramientas de organización como pueden ser las propias tablas de Gitlab o TeamGantt.

1.5 Fases del proyecto

Una vez presentados los principales objetivos que tratan de lograrse a través del proyecto, posteriormente se procederá a concretar las principales fases a través de los cuales debe desenvolverse el mismo.

El siguiente diagrama está incluido en el anexo, por si se quisiera apreciar con mayor detalle:

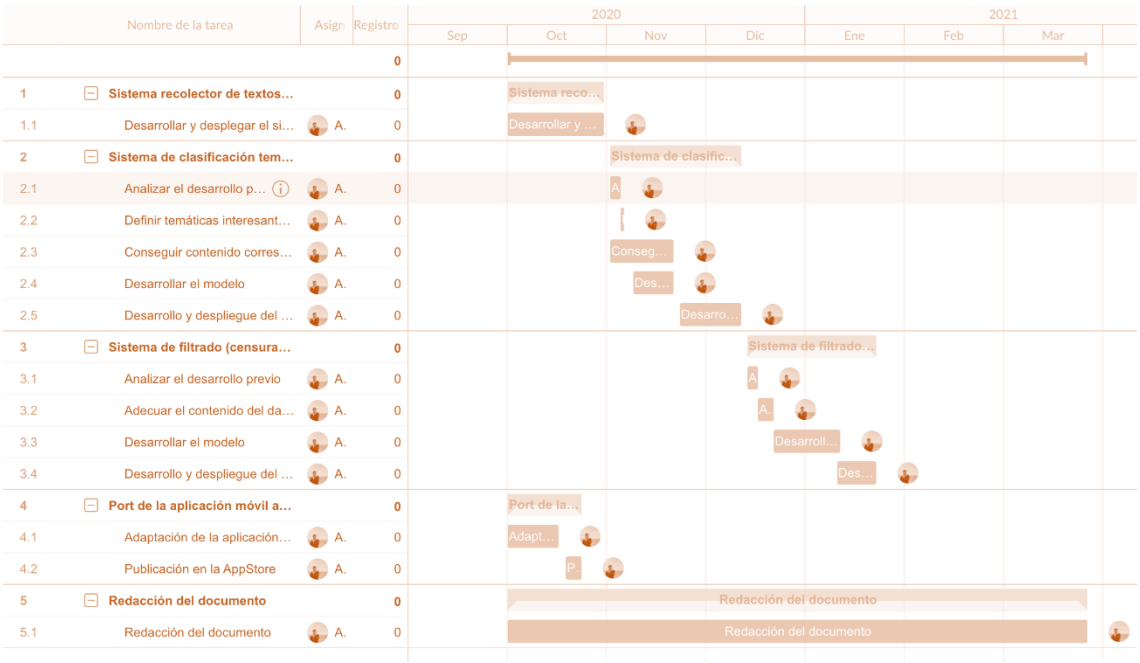


Ilustración 2: Diagrama de Gantt del proyecto

Como se puede observar, se han diferenciado 4 fases más una extra que recorrerá todo el transcurso del proyecto (la redacción del informe). A continuación, se explicará en que consiste cada una de las fases en orden cronológico.

1.5.1 Fase 1 + 4: Desarrollo del software recolector + Adaptación y publicación de la aplicación Tentu a iOS

Esta fase ha transcurrido al comienzo del desarrollo del proyecto, concretamente en el mes de octubre. Esto tiene un motivo y no es otro que desarrollar y lanzar cuanto antes el software recolector con el fin de que recopile datos autónomamente mientras el equipo de desarrollo se centra en otras tareas como la investigación del modelo.

Para desarrollar dicho software, es necesario investigar con qué recursos en la nube cuenta la empresa, por si alguno fuera reutilizable y ventajoso para el desarrollo.

Por otro lado, se debe analizar cuál puede ser la plataforma de despliegue que mejor se adapte a las necesidades de la empresa tanto tecnológica como económicamente.

Dicha investigación y toma de decisiones puede verse en el apartado 2.1.6 de este mismo documento.

Como se ha indicado en la contextualización, el hecho de llevar varios años en la empresa conlleva que colabore en otros desarrollos como los de la revista Tentu. En este caso esta fase también va a compaginar lo explicado hasta ahora, con la adaptación de la aplicación Android al sistema operativo de los dispositivos de Apple, iOS.

1.5.2 Fase 2: Desarrollo del sistema de clasificación automática

Tras desarrollar y desplegar la aplicación recolectora, la labor de esta fase consiste en desarrollar y ofrecer un sistema de clasificación de contenidos a los clientes.

Para ello, lo primero que se debe hacer en esta fase es identificar las temáticas que más interesantes pueden resultar a los clientes. Tras identificar varias temáticas se deben buscar fuentes RSS correspondientes a dichas categorías e introducirlas en el software con el fin de que busque artículos.

Mientras el software recopilador trabaja, se debe investigar a cerca del tratamiento que se le debe dar a los datos y que algoritmos pueden resultar adecuados para la clasificación de textos multiclase. Una vez investigado, se tratará de implementarlo en un entorno de entrenamiento en la nube.

Después de conocer el algoritmo que mejor resultado proporciona con los datos recopilados, se tratará de buscar la forma adecuada de proporcionar predicciones del modelo realizado a los clientes.

1.5.3 Fase 3: Desarrollo del sistema de filtrado automático

Es cuando termina la labor de la fase dos cuando empieza la de la tercera. Esta conlleva realizar un estudio sobre la realización de un modelo que sea capaz de proporcionar un grado de censura del texto recibido.

Todo empieza con la búsqueda de posibles fuentes donde encontrar textos malsonantes con los que poder formas un conjunto de datos de entrenamiento. Tras reunir suficientes datos, al igual que en el desarrollo del sistema de clasificación automática, se debe investigar a cerca del tratamiento que se le debe dar a los datos y que algoritmos pueden resultar adecuados.

Finalmente, se desplegará de forma que sea sencillo proporcionar predicciones del modelo a los clientes.

1.5.4 Fase 5: Redacción del informe

Como se puede ver en el diagrama de Gantt, esta fase va a transcurrir a lo largo de todo el proyecto. El principal motivo de ello es que a medida que se vaya desarrollando el proyecto, se irán anotando diferentes detalles a modo de diario con el fin de que no se olvide nada la hora de desarrollar el documento.

1.6 Pliego de condiciones

En el siguiente apartado, se presentan los pliegos de desarrollo y ejecución del proyecto.

1.6.1 Pliego de condiciones técnicas de desarrollo

En este apartado, se definirán tanto los equipos, como los softwares necesarios para el correcto desarrollo del proyecto.

1.6.1.1 Especificaciones de equipos

Será necesario un equipo con las siguientes características mínimas:

- Procesador Intel Core i3 de 2 núcleos y 2 hilos.
- Memoria RAM de 4GB DDR3.
- Disco duro de 125 GB.
- Sistema Operativo Windows 10 / MAC / Linux.

Las especificaciones del equipo con las que se está desarrollando el proyecto son las siguientes:

- Procesador Intel Core i5 8250U @ 2.30GHz de 4 núcleos y 8 hilos.
- Memoria RAM de 8GB LPDDR3 a 2133 MHz.
- Disco duro SSD de 256 GB.
- Sistema Operativo macOS Catalina.

1.6.1.2 Especificaciones de software

El software necesario para el desarrollo del proyecto es el siguiente:

- Visual Studio Code para el desarrollo del código del proyecto.
- React JS, una librería para crear interfaces de usuario de una aplicación web.
- Postman para la comprobación del funcionamiento de los servicios REST desarrollados.
- Keras para el desarrollo de diferentes modelos de inteligencia artificial.
- Flutter para el desarrollo de aplicaciones móviles, tanto para iOS como Android.

1.6.1.3 Especificaciones de calidad de software

Se debe de tener en cuenta que, a la hora de desarrollar el producto, este debe cumplir unos mínimos estándares de calidad, así como unas normas de codificación concretas, con el fin de conseguirlo, se utilizarán las siguientes herramientas:

- JavaScript Standard Style: Adaptarse a un estilo estándar supone adoptar convenciones de la comunidad más que el estilo personal con el fin de que el código sea más claro.
- GitLab: Se trata de una herramienta de gestión de código que además también se encarga de la integración continua, de modo que cada vez que se realiza una actualización del código almacenado (commit) se ejecutan los diferentes testeos previamente configurados, así como el despliegue de la aplicación.

2 Desarrollo

A lo largo de este apartado se hará una descripción de las tareas realizadas. Empezando por el software recopilador, siguiendo con el desarrollo/investigación del modelo y terminando con los desarrollos adicionales. En cada uno de los apartados se detallará tanto la toma de decisiones como el desarrollo llevado a cabo. Empecemos con el software recopilador

2.1 Software recopilador

En este apartado se comenzará con una contextualización y un análisis de los recursos necesarios. Teniendo en cuenta esto, se detallará una arquitectura y su funcionamiento y finalmente se explicará el procedimiento seguido para su despliegue.

2.1.1 Contexto

Como principalmente se trabajará con algoritmos de aprendizaje supervisado, los modelos necesitan ser entrenados con una gran cantidad de datos. Estos, deben disponer de una categoría concreta que quiera ser introducida en el sistema, para que posteriormente este pueda reconocerlos.

En cuanto a la fuente de datos sobre los cuales entrenar el modelo, serán obtenidos mediante fuentes RSS que dispongan de una categoría fija, después de una revisión para comprobar que los contenidos disponibles pertenecen realmente a esa categoría. Estos serán recogidos con un sistema crawling a desarrollar, entrando en cada uno de los enlaces disponibles en la fuente RSS y obteniendo los datos relevantes, como el texto del contenido o su título mediante el uso de códigos XPath.

Cabe destacar la importancia de este software recopilador en varios aspectos. Por un lado, resulta de vital importancia que este se encuentre siempre disponible, ya que el objetivo es que sea totalmente autónomo y no requiera de mantenimiento por parte del desarrollador.

Igualmente, sería conveniente que, en la medida de lo posible, no suponga ningún gasto económico adicional a la empresa, sacando el máximo partido a los servicios existentes en la organización, por lo que se debe analizar que ofrece cada proveedor en su capa de uso gratuita.

Por otro lado, es conveniente que el accionado de este software sea automático de modo que no tenga que ser accionado manualmente. De esta forma, también se asegura que se ejecute mínimamente una vez al día, no dando lugar a posibles descuidos humanos.

Además, sería adecuado que este software hiciese un mínimo preprocesamiento de los datos recopilados, restando trabajo al desarrollador que haga uso de ellos. Por lo que se debe investigar que preprocesamiento se suele seguir en este tipo de desarrollos.

Asimismo, al ser un software recopilador, resulta fundamental poder modificar el listado de fuentes a revisar en cada llamada sin tener que actualizar el repositorio de código.

Igualmente, es importante asegurar que el software vaya a funcionar correctamente sea cual sea la plataforma de despliegue, restando así, importancia a la misma.

En resumidas cuentas, se debe conseguir que la aplicación sea totalmente autónoma, sin requerir ninguna atención por parte del desarrollador. Además, debe ser sencillo modificar el listado de fuentes a consultar y que sea el mismo software quien haga parte del preprocesamiento de los datos.

A lo largo de este apartado se detallará el desarrollo llevado a cabo para lograr lo explicado. Se empezará con la identificación de los recursos necesarios y la valoración de los disponibles.

2.1.2 Análisis de los recursos necesarios para el desarrollo

Antes de comenzar a desarrollar se deben de identificar los recursos informáticos que serán necesarios, así como con los que cuenta la empresa, por si alguno de ellos fuera aprovechable.

En este caso, los recursos necesarios identificados son principalmente una plataforma de despliegue en la que alojar la aplicación y un almacenamiento en la nube donde tener guardados los resultados y el listado de fuentes a consultar.

Justamente ISEA cuenta con Firebase para el desarrollo de GAIA y Tentu. Firebase es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo.

Aunque fue creada en 2011 pasó a ser parte de Google en 2014, comenzando como una base de datos en tiempo real. Sin embargo, se añadieron más y más funciones que, en parte, permitieron agrupar los SDK de productos de Google con distintos fines, facilitando su uso.

Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar los ficheros de sus aplicaciones (y vinculándolos con referencias a un árbol de ficheros para mejorar el rendimiento de la aplicación) y sincronizarlos.

Este almacenamiento puede ser de gran ayuda para almacenar tanto los resultados obtenidos como el listado de fuentes a consultar en cada una de las ejecuciones. Además, al contar con experiencia trabajando con esta herramienta, parece la adecuada para el almacenamiento.

Por lo que solamente faltaría identificar una plataforma adecuada de despliegue para la aplicación, de la que se hará una comparativa y selección en el apartado 2.1.6.

Tras identificar los recursos necesarios y los disponibles por parte de la empresa, se procederá a plantear una arquitectura para dar solución al problema.

2.1.3 Arquitectura propuesta para el recopilador

La arquitectura propuesta para dar solución al problema es la siguiente:

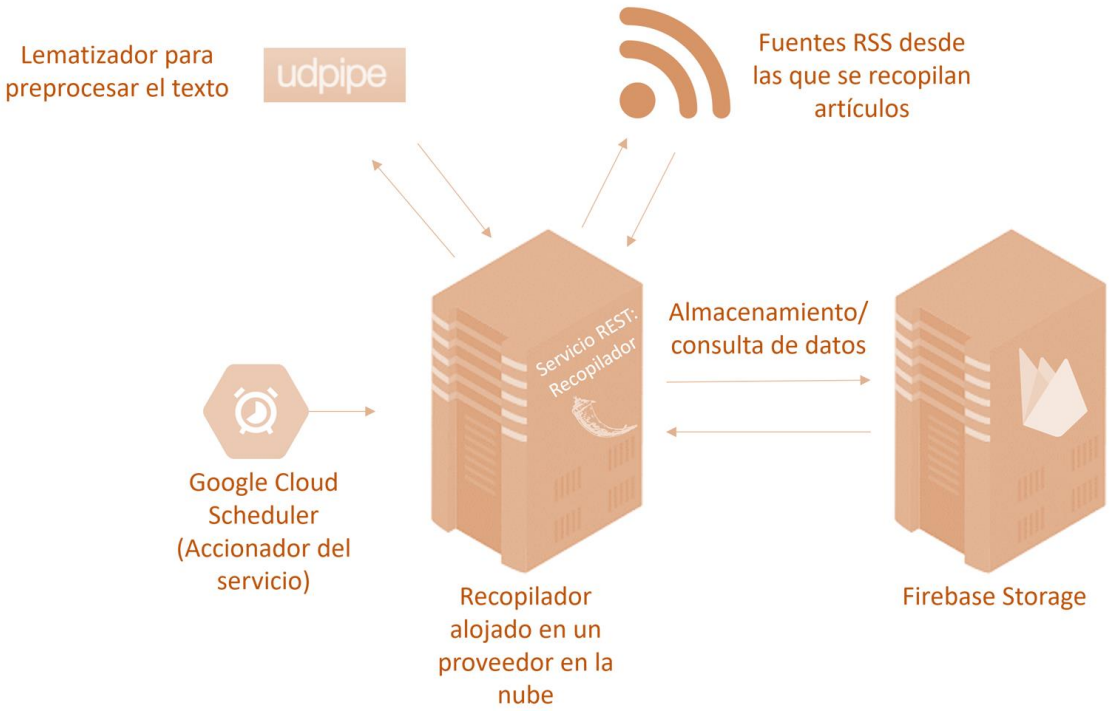


Ilustración 3: Arquitectura propuesta para el recopilador

En ella se pueden diferenciar dos actores principales, por un lado, un servicio REST realizado mediante Flask que se encarga de llevar la lógica principal de la aplicación y por otro Google Firebase Storage que se ocupa de almacenar los datos recopilados por el software.

Cabe destacar, que con el fin cumplir uno de los retos con relación al accionado automático del servicio, se ha decidido utilizar Google Cloud Scheduler. De este modo, se asegura que el servicio se ejecute mínimamente una vez al día, no dando lugar a posibles descuidos humanos.

En el siguiente apartado se detallará técnicamente el funcionamiento interno de cada uno de los componentes del software, empezando por el procesamiento de datos que realiza la aplicación REST antes de almacenarla en Firebase Storage.

2.1.4 Preprocesamiento de los datos realizado por parte del recolector

Como se ha indicado en la contextualización, sería adecuado que el software fuera capaz de realizar un preprocesamiento de los datos. En este caso, se ha seguido un preproceso muy concreto para problemas de clasificación de textos. Cabe destacar que se ha realizado una búsqueda sobre cuál es el tratamiento de datos más eficaz para los problemas basados en NLP (Natural Lenguage Processing), llegando a la conclusión desarrollada en esta sección. La labor realizada se basa principalmente en el libro Tecnologías del lenguaje [10].

El esquema propuesto en el libro para el análisis textual es el siguiente:

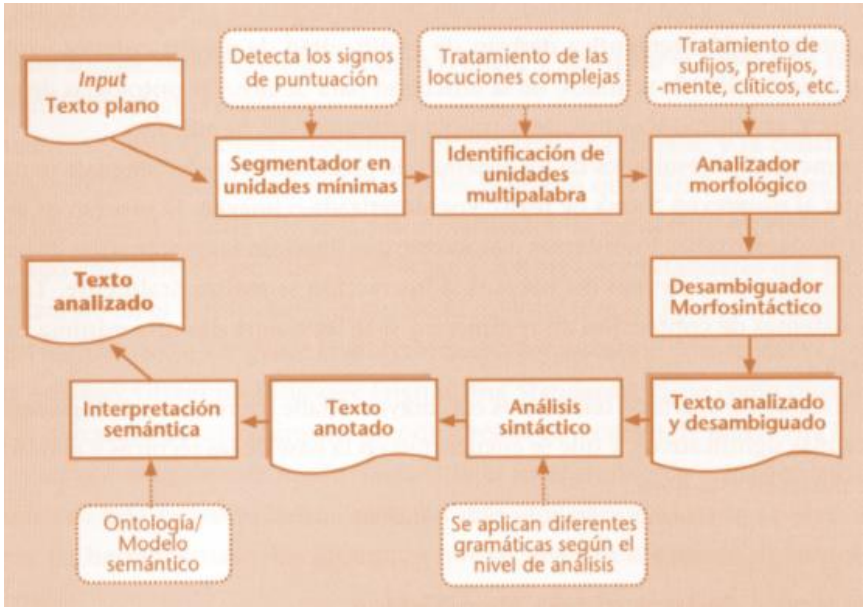


Ilustración 4: Esquema propuesto en el libro "Tecnologías del lenguaje"

Aunque no se ha llegado a la profundidad alcanzada por el libro ya que, desde el punto de vista del equipo de desarrollo, la relación esfuerzo/recompensa era baja. Dicho esto, el preproceso realizado por el software recopilador consiste principalmente en lo siguiente.

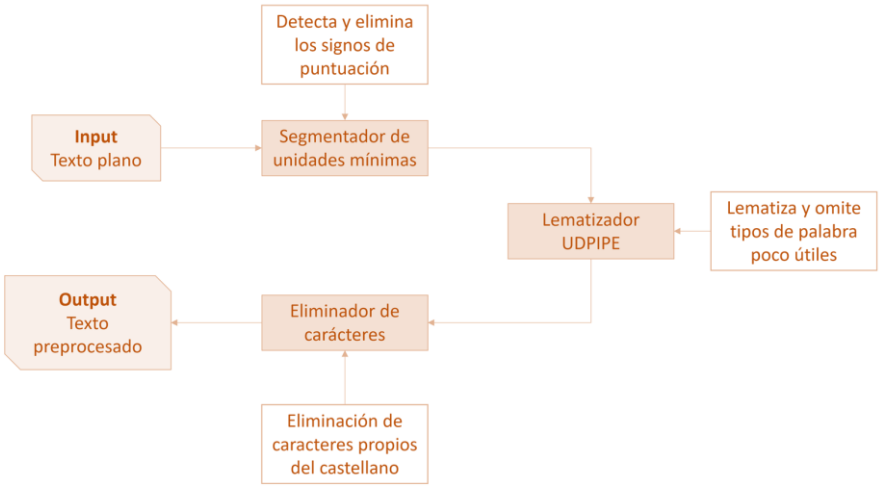


Ilustración 5: Esquema seguido para el preproceso de los artículos recopilados

Después de recoger estos datos y antes de almacenarlos para su posterior uso en el modelo, se procede a realizar la lematización del contenido. Además de ello, un filtrado es aplicado para eliminar el texto no deseado para su posterior uso en el modelo, ya que no aporta en la categorización de los contenidos.

La lematización, es un proceso lingüístico que consiste en, dada una forma flexionada (es decir, en plural, en femenino o masculino, conjugada…) de una palabra, hallar el lema correspondiente, es decir, la forma que se acepta como representante de todas las formas flexionadas de la misma, en otras palabras, el núcleo. En anteriores casos se ha utilizado el lematizador ofrecido por UZEI, pero en esta ocasión se utilizará una herramienta de uso gratuito llamado UDPipe [11].

UDPipe, ofrece, entre otros, servicios de lematización en una gran cantidad de idiomas y será utilizado en reemplazo al sistema de UZEI para intentar construir un sistema completo sin tener que depender de sus servicios y productos. Por otra parte, el lematizador de UDPipe, al estar disponible en una gran variedad de idiomas puede ser útil para el proyecto en un futuro, ofreciendo posibilidades de expansión.

Además de lematizar el texto, el servicio de UDPipe, al igual que el de UZEI, ofrecen una gran variedad de datos lingüísticos como el tipo de palabra (determinante, nombre, verbo…) y un análisis morfológico del mismo (genero, plural/singular, persona…) tal y como se puede ver en la siguiente ilustración:

Id	Form	Lemma	UPosTag	XPosTag	Feats	Head	DepRel	Deps	Misc
# text = Los desarrolladores ya pueden probar la nueva versión del sistema operativo para iPhone que se presentó este lunes en la conferencia anual para desarrolladores de Apple.									
1	Los	el	DET	_	Definite=Def|Gender=Masc|Number=Plur|PronType=Art	2	det	_	_
2	desarrolladores	desarrollador	NOUN	_	Gender=Masc|Number=Plur	5	nsubj	_	_
3	ya	ya	ADV	_	_	5	advmod	_	_
4	pueden	poder	AUX	_	Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin	5	aux	_	_
5	probar	probar	VERB	_	VerbForm=Inf	0	root	_	_
6	la	el	DET	_	Definite=Def|Gender=Fem|Number=Sing|PronType=Art	8	det	_	_
7	nueva	nuevo	ADJ	_	Gender=Fem|Number=Sing	8	amod	_	_
8	versión	versión	NOUN	_	Gender=Fem|Number=Sing	5	obj	_	_
9-10	del	_	_	_	_	_	_	_	_
9	de	de	ADP	_	_	11	case	_	_
10	el	el	DET	_	Definite=Def|Gender=Masc|Number=Sing|PronType=Art	11	det	_	_
11	sistema	sistema	NOUN	_	Gender=Masc|Number=Sing	8	nmod	_	_
12	operativo	operativo	ADJ	_	Gender=Masc|Number=Sing	11	amod	_	_
13	para	para	ADP	_	_	14	case	_	_
14	iPhone	iphone	PROPN	_	Gender=Masc|Number=Sing	8	nmod	_	_

Ilustración 6: Ejemplo de uso de UDPIPE

Aprovechando que el servicio provee esos datos adicionales, la característica “tipo de palabra” será utilizada para excluir ciertas palabras a la hora de crear el corpus, puesto que algunos tipos de palabras, como por ejemplo los determinantes o las preposiciones, no proveen información relevante a la categoría a la que están relacionadas.

Para ello, se realiza una petición por cada contenido y se analiza la respuesta. Dicho análisis tiene que ser realizado automáticamente, ya que se trata de una gran cantidad de documentos. A tales efectos y con el propósito de recoger tan solo los datos relevantes de la respuesta recibida, se ha creado una expresión regular (Regex), que recoge el lema de cada palabra.

Además de ello, se realizará una limpieza de las palabras eliminando cualquier tipo de acentuación o caracteres especiales de la lengua española, como las “ñ” o las “ç”. El propósito es evitar problemas en caso de que el texto venga con alguna falta ortográfica de ese tipo y así evitar confusiones.

Por último, estos datos deberán de ser almacenados en un fichero con el fin de poder utilizarlos tanto en la fase de entrenamiento de los modelos como en la fase de validación. Quedando de la siguiente forma:

418	Carne membrillo Thermomix rapido limpio facil The	food	2020-11-05T14:03:49
419	Joeca chorizo guiso patata tipico aguilar frontera pu	food	2020-11-05T14:03:48
420	galleta chocolate aceite oliva virgen extra bocado ri	food	2020-11-05T14:03:49
421	llanda almendra torto boba coca llanda llanda lata	food	2020-11-05T14:03:49
422	Ajilimojili tipico jaen elaboracion tradicional cocina	food	2020-11-05T14:03:49
423	Coquina huelva ajillo vino blanco receta rapido facil	food	2020-11-05T14:03:49
424	caballla moruna receta tradicional ceuta receta fac	food	2020-11-05T14:03:49
425	dia septiembre final concurso tapas Gastronomicas	food	2020-11-05T14:03:49
426	Bollito leche aove chocolate aceite oliva virgen extr	food	2020-11-05T14:03:49
427	plato tipico gastronomia canario gofio escaldado g	food	2020-11-05T14:03:48
428	seguro Instagram foto tentadora cuenco fruta batid	food	2020-11-05T14:03:48
429	receta bizcocho casero facil rico mejor aroma bizco	food	2020-11-05T14:03:48
430	momento dificil tiempo casa mejor comida cena fa	food	2020-11-05T14:03:47

Ilustración 7: Captura del fichero CSV con los datos recogidos por el recopilador

En gran parte, la validez del modelo dependerá de la calidad de los datos que se hayan recopilado para el entrenamiento y por lo tanto es de gran importancia que los datos sean obtenidos de fuentes fiables y estén clasificados correctamente. Por eso se le dará una gran importancia a la revisión de las fuentes RSS antes de su inclusión en el listado del apartado 2.1.5.1.

Tras concluir con el preprocesamiento de los datos, se explicará el funcionamiento interno del servicio REST en sí, es decir, que lógica sigue la aplicación cada vez que es accionado por Google Cloud Scheduler.

2.1.5 Funcionamiento del recopilador

En este apartado se explicará la lógica detrás del servicio REST, pero para ello, resulta imprescindible la comprensión de los ficheros que hacen posible su funcionamiento.

2.1.5.1 Listado de fuentes RSS

Cabe destacar que la recolección de los datos se realizará en base a fuentes RSS que estén directamente relacionadas con una categoría especifica. Varios medios de comunicación españoles disponen de ellos en sus sitios web, como por ejemplo El Correo, tal y como se puede ver en la imagen:

Ilustración 8: Ejemplo con las fuentes RSS ofrecidas por El Correo

Tras verificar que un RSS dispone de calidad suficiente para ser añadido al listado, se le asociará un código XPATH con el fin de recoger tan solo la información relevante. De esta forma, se excluyen las imágenes o los típicos enlaces para compartir las noticias a través de las habituales redes sociales, ya que estos no aportan datos diferenciables de cada categoría y harían que el resultado del modelo empeorase significativamente.

Este proceso de recolección de fuentes es realizado de forma completamente manual. La recolección es reflejada en un fichero JSON, conteniendo un objeto en el que cada una de las categorías tendrá una lista con cada uno de los RSS de los que se recogerán datos, además del código XPATH correspondiente a esa fuente con el fin de obtener tan solo el contenido deseado, tal y como se puede ver en la ilustración:

```
{
  "food":[
    {
      "url":"https://elcomidista.elpais.com/rss/elcomidista/portada.xml",
      "x":"//div[@class='cuerpo']"
    },
    {
      "url":"https://www.javirecetas.com/feed/",
      "x":"//div[@class='post']"
    },
    {
      "url":"https://webosfritos.es/feed/",
      "x":"//div[@class='pf-content']"
    },
    {
      "url":"https://gastronomiaycia.republica.com/feed/",
      "x":"//div[@itemprop='articleBody']"
    },
```

Ilustración 9: Ejemplo de la relación creada entre la fuente RSS y su código XPATH

2.1.5.2 Hash set

Con el fin de no malgastar el tiempo en llamadas repetidas en cada ejecución del recopilador, se ha decidido almacenar en Google Firebase Storage un listado con un historial de las páginas visitadas.

Este listado es actualizado en la finalización de cada una de las ejecuciones.

2.1.5.3 Lógica detrás del servicio

La lógica detrás del servicio se ha diferenciado en tres fases, resulta de interés indicar que cada una de las fases transcurren de forma secuencial, tras el accionado automático de Google Cloud Scheduler:

- Fase 1: Es la encargada de iniciar la sesión con los servicios de Google, en este caso Firebase Storage y conseguir los ficheros targets.json y hash_set, totalmente imprescindibles para el funcionamiento de la aplicación.
- Fase 2: Su principal labor consiste en recorrer la lista de fuentes RSS en busca de nuevos artículos que añadir al conjunto de datos. Para ello, antes de añadirlo le hace un pequeño preproceso tal y como se ha explicado en el apartado 2.1.4 de este mismo documento. Una vez se tiene el contenido preprocesado se añade su hash al historial con el fin de que no sea duplicado en el conjunto de datos final.
- Fase 3: Tras recorrer todo el listado de fuentes, la labor de esta fase se centra en la subida de datos a Google Firebase Storage. Tras esta fase, se da por concluida la ejecución del recopilador.

Para ayudar a la comprensión se ha realizado el siguiente diagrama de secuencia:

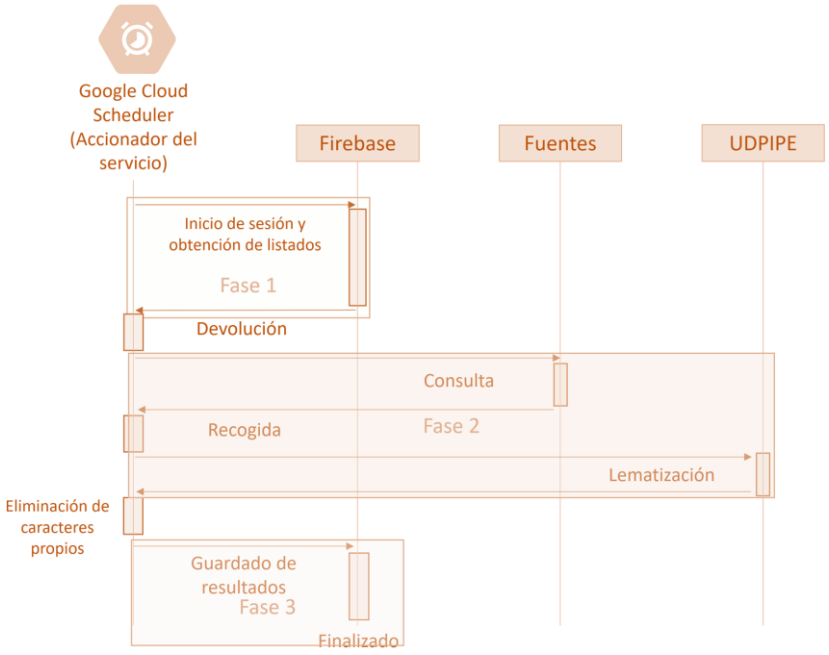


Ilustración 10: Diagrama de secuencia sobre el funcionamiento del recopilador

Por otro lado, al ser un software que está destinado a estar en producción, es recomendable hacer uso de un WSGI [12]. Pese a que tanto Django como Flask en este caso tienen servidores web integrados que se suelen usar para pruebas, sólo se recomienda el uso de estos precisamente para eso: entornos de pruebas. Al poner un software en producción, generalmente se usa Nginx para hacer de proxy inverso y Gunicorn como servidor WSGI. Esto proporciona diferentes beneficios como los siguientes:

- Ocultar información (versión del servidor web…)
- Se pueden enlazar conexiones https/SSL y asignarlos a los servicios http internos.
- Centralizar todos los DDOS de prevención

Gunicorn puede escalar con varios workers trabajando en paralelo y conectándose con la aplicación Django/Flask. Quedaría de la siguiente forma:

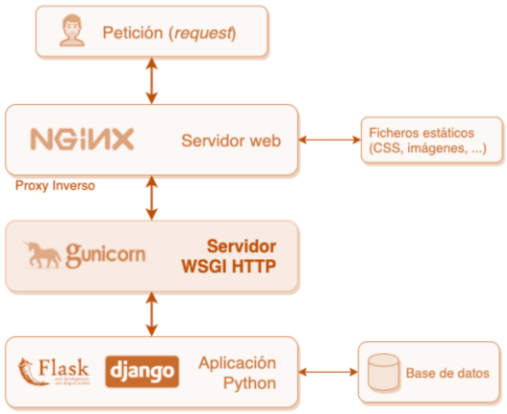


Ilustración 11: Esquema de funcionamiento WSGI

Por tanto, Gunicorn aporta, entre otras cosas, escalabilidad y la seguridad de que la aplicación va a estar constantemente disponible de cara al cliente ya que, si por algún motivo no contemplado la aplicación deja de responder, Gunicorn se encargará de reinicializarla.

Tras explicar el funcionamiento de la aplicación se procederá a detallar el despliegue de este.

2.1.6 Despliegue y puesta en marcha del software

Como se ha indicado en la contextualización al comienzo de la sección, es importante asegurar que el software pueda funcionar sea cual sea el proveedor de despliegue. Una de las opciones que más interesantes parecen en el mercado actual es Docker, ya que proporciona mucha flexibilidad y comodidad al desarrollador que lo utiliza.

En el siguiente apartado se analizará esta opción en profundidad.

2.1.6.1 ¿Qué es Docker?

La idea detrás de Docker es crear contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado,

independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

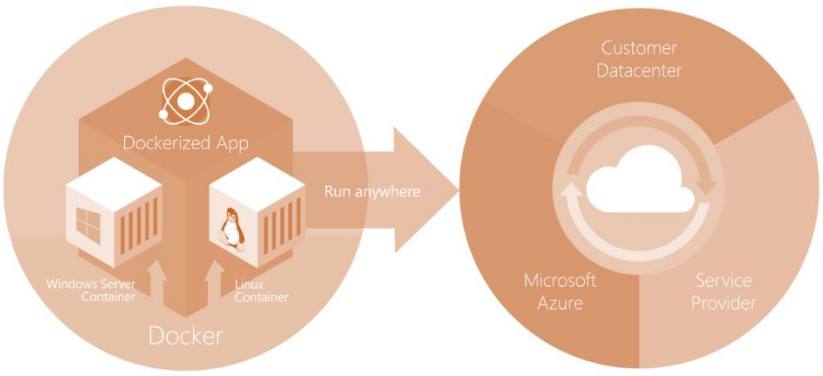


Ilustración 12: Ejemplo de que una aplicación Dockerizada se puede desplegar en prácticamente cualquier proveedor

Para poder acceder como usuarios normales a una aplicación, dicho software necesita estar ejecutándose en un equipo. Además, dependiendo del tipo de aplicación, el equipo también necesita tener instaladas una serie de dependencias para que la aplicación se ejecute correctamente: una versión concreta de Java instalado o un servidor de aplicaciones como por ejemplo Tomcat.

Docker, permite meter en un contenedor todas aquellas dependencias que la aplicación necesita para ser ejecutada (Java, Maven, Tomcat…) además del código fuente de la propia aplicación. De este modo, la aplicación es ejecutable en cualquier equipo con Docker instalado, sin tener que preocuparse por nada más.

Resumiendo, el objetivo de Docker es ejecutar la aplicación software desde el contenedor. Un contenedor no es otra cosa que una agrupación del código fuente de la aplicación a ejecutar y todas sus dependencias.

2.1.6.1.1 ¿Qué beneficios tiene?

Los beneficios de Docker son varios, pero estos son los más destacables:

- Modularidad: El enfoque Docker para la creación de contenedores se centra en la capacidad de tomar una parte de una aplicación, para actualizarla o repararla, sin necesidad de tomar la aplicación completa. Además, este enfoque basado en los microservicios puede compartir procesos entre varias aplicaciones de la misma forma que funciona la arquitectura orientada al servicio (SOA).
- Control de versiones y capas: Cada Dockerfile (fichero con el que se forman las imágenes con las que se componen los contenedores) se compone de una serie de capas. Estas capas se combinan en una sola imagen. Una capa se crea cuando la imagen cambia. Cada vez que se especifica un comando, como ejecutar o copiar, se crea una nueva capa.

Docker reutiliza estas capas para construir nuevos contenedores, lo cual hace mucho más rápido el proceso de construcción. Los cambios intermedios se comparten entre imágenes, mejorando aún más la velocidad, el tamaño y la eficiencia. El control de versiones es inherente a la creación de capas. Cada vez que se produce un cambio nuevo se tiene un control completo de las imágenes de contenedor.

- Restauración: Probablemente la mejor parte de la creación de capas es la capacidad de restaurar o recuperar. Todas las imágenes de Docker están construidas por diferentes capas. Si por algún motivo la imagen actual no resulta válida (puede que se haya encontrado algún bug, por ejemplo) es posible restaurarla a la versión anterior. Esto es compatible con un enfoque de desarrollo ágil y permite hacer realidad la integración e implementación continuas (CI/CD) desde una perspectiva de las herramientas.
- Implementación rápida: Antes el hecho de montar, ejecutar, proveer y facilitar era una cuestión de días o incluso semanas y el nivel de esfuerzo y sobrecarga era extenuante. Hoy en día, con los contenedores basados en Docker se puede reducir el tiempo de implementación a segundos. Al crear un contenedor para cada proceso, se puede compartir rápidamente los procesos similares con nuevas aplicaciones. Y, debido a que un SO no necesita iniciarse para agregar o mover un contenedor, los tiempos de implementación son sustancialmente inferiores en comparación a las máquinas virtuales convencionales. Además, con la velocidad de implementación, se puede crear y destruir la información creada por sus contenedores sin preocupación, de forma fácil y rentable. Por lo tanto, la tecnología Docker es un enfoque más granular y controlable, basado en microservicios, que prioriza la eficiencia.

2.1.6.1.2 Diferencias con una máquina virtual convencional

Una de las principales dudas es en qué se diferencia entonces un contenedor software y una máquina virtual. De hecho, este concepto es mucho más anterior que el de los propios contenedores.

Gracias a la virtualización es posible, usando un mismo ordenador, de tener distintas máquinas virtuales con su propio sistema operativo, Linux o Windows. Todo ello ejecutándose en un sistema operativo anfitrión y con acceso virtualizado al hardware.

La virtualización es una práctica habitual en servidores para alojar diferentes aplicaciones o en nuestro propio entorno de trabajo para ejecutar distintos sistemas operativos, por ejemplo. Muchos alojamientos de hosting tradicionales se han basado en crear máquinas virtuales limitadas sobre el mismo servidor para alojar nuestros servidores web de forma aislada, siendo compartido por una decena de clientes.

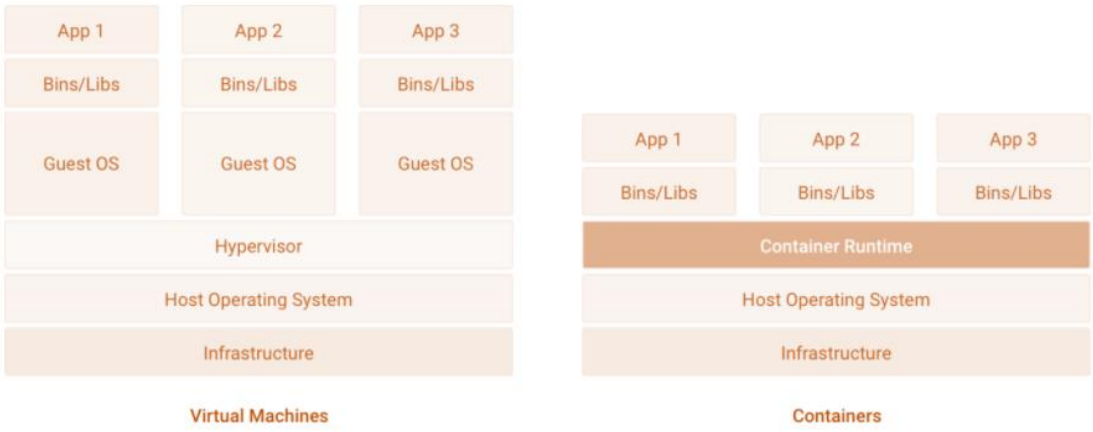


Ilustración 13: Comparación entre el funcionamiento de un contenedor Docker y una máquina virtual corriente

En comparación a las máquinas virtuales, los contenedores se ejecutan sobre el mismo sistema operativo host de forma aislada también, aunque sin necesidad de un sistema operativo propio, ya que comparten el mismo Kernel, lo que los hace mucho más ligeros. De donde se pueden sacar 3 máquinas virtuales probablemente se pueda multiplicar por un gran número de contenedores software.

Un contenedor de Docker puede ocupar tan solo unas cuantas decenas de megas mientras que una máquina virtual, al tener que emular todo un sistema operativo, puede ocupar varias gigas de memoria. Lo cual representa un primer punto en el ahorro de coste.

Habitualmente cada aplicación en Docker va en su propio contenedor totalmente aislado, mientras que en las máquinas virtuales es habitual debido al dimensionamiento tener varias aplicaciones en la misma máquina con sus propias dependencias, mucho peor para escalar de forma horizontal.

Tras argumentar por que se ha elegido Docker como herramienta para el despliegue, se analizarán diferentes proveedores en los que poder desplegar la aplicación contenedorizada.

2.1.6.2 Selección de proveedor de despliegue

Debido a la popularidad de las aplicaciones contenedorizadas, hoy en día existen multitud de proveedores que posibilitan el despliegue de este tipo de aplicaciones, entre las que se encuentran Microsoft Azure, Amazon Web Services o Heroku.

En este caso al tener experiencia de desarrollos anteriores en los proveedores Amazon y Heroku, serán los principales proveedores por comparar.

2.1.6.2.1 Heroku

Heroku es una plataforma como servicio (PaaS) de computación en la Nube que soporta distintos lenguajes de programación.

Heroku es propiedad de Salesforce.com. Es una de las primeras plataformas de computación en la nube, que fue desarrollada desde junio de 2007, con el objetivo de soportar solamente el lenguaje de programación Ruby, pero posteriormente se ha extendido el soporte a Java, Node.js, Scala, Clojure, Python, PHP, Go y el despliegue a través de contenedores Docker, que es la parte que interesa para el desarrollo del proyecto [13].

Las especificaciones que tiene la máquina en el rango gratuito según la propia página de Heroku son las siguientes:

Dyno Type	Memory (RAM)	CPU Share	Compute	Dedicated	Sleeps
free	512 MB	1x	1x-4x	no	yes
hobby	512 MB	1x	1x-4x	no	no
standard-1x	512 MB	1x	1x-4x	no	no
standard-2x	1024 MB	2x	4x-8x	no	no
performance-m	2.5 GB	100%	12x	yes	no
performance-l	14 GB	100%	50x	yes	no

Tabla 1: Rango de precios Heroku

Como se ha indicado anteriormente, uno de los objetivos es intentar que este sistema se mantenga operativo sin suponer ningún tipo de coste adicional (más allá del almacenamiento de los datos) a la empresa. Es por ello por lo que el equipo que corresponde es el de 512MB de memoria RAM y una CPU. Además, este equipo en periodos de inactividad no se quedaría disponible para el cliente y la primera ejecución tras la hibernación conllevaría un alto tiempo de respuesta, aunque justamente en un proyecto como este esa peculiaridad no es de relevancia.

Tras hacer una primera prueba con Heroku, se ha comprobado que los 512MB de RAM son totalmente insuficientes para el correcto despliegue de la aplicación, y que debido al WSGI, no hace nada más que reiniciarse la inicialización de la aplicación. Es por ello por lo que se decidió analizar la opción de Amazon Web Services.

2.1.6.2.2 Amazon Web Services

AWS es una colección de servicios de computación en la nube pública (también llamados servicios web) que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com. Es usado en aplicaciones populares como Dropbox, Foursquare, HootSuite. Además, es una de las ofertas internacionales más importantes de la computación en la nube y compite directamente contra servicios como Microsoft Azure, Google Cloud Platform e IBM Cloud. Es considerado como un pionero en este campo.

Amazon EC2 es una parte central de la plataforma de cómputo en la nube de AWS [14]. EC2 permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones. Este tipo de servicio supone un cambio en el modelo informático al proporcionar capacidad informática con tamaño modificable en la nube, pagando por la

capacidad utilizada. En lugar de comprar o alquilar un determinado procesador para utilizarlo varios meses o años, en EC2 se alquila la capacidad por horas.

En este caso, el producto en el que se va a intentar desplegar el contenedor de Docker es EC2, aunque existen alternativas más preparadas para los contenedores como AWS ECS. El principal motivo por el que se va a ignorar de momento ECS es el desconocimiento del producto y la urgencia de desplegarlo cuanto antes con el fin de que el software se disponga a realizar su labor, recopilar artículos.

Esto no quiere decir que el producto se vaya a ignorar por completo como se puede observar en el apartado de desarrollos adicionales.

EC2 ofrece un gran rango de precios dependiendo de la máquina seleccionada, en este caso, al igual que Heroku, AWS ofrece una capa gratuita con una máquina de tipo *t2.micro* con un único núcleo y lo que es más importante, 1 GB de RAM que es prácticamente el doble de lo que ofrece Heroku.

Tras alquilar una instancia EC2, al estar la aplicación contenedorizada, es realmente sencillo desplegarla ya que únicamente es necesario instalar Docker. Tras iniciarlo y comprobar que todo funciona correctamente mediante una petición manual de Postman, se ha automatizado la petición a la aplicación con Google Cloud Scheduler. De este modo, ninguno de los desarrolladores debe estar pendiente de si el sistema funciona o no, solamente debe recoger los datos en el momento en el que le interese trabajar con ellos.

2.1.6.3 Integración y despliegue continúa

Por motivos que se explicarán en el apartado 2.4.1 de este informe, la integración y despliegue continúa juegan un papel muy importante en el desarrollo de los softwares de hoy en día. Especialmente en aquellos que consisten en la entrega continua de un producto a los clientes, con el fin de evitar posibles fallos en el software en producción.

Como este no es el caso y el software solamente está diseñado para tener un uso interno, se ha decidido no invertir tiempo en implementar ningún tipo de integración y despliegue automático al menos en el recopilador. Como se podrá comprobar en siguientes apartados, existen diferentes desarrollos que si se han adaptado para tener integración y despliegue continúo.

2.1.7 Finalización del recopilador

Una vez finalizado el desarrollo del recopilador y tras tenerlo en funcionamiento durante algunos meses, el sistema desplegado es capaz de alcanzar un volumen de en torno a 2000-2500 artículos recopilados de forma diaria. Cabe destacar que la aplicación no tiene ningún tipo de límite y que cuantas más fuentes se le introduzcan, el volumen de artículos irá aumentando.

Además, resulta sencillo modificar el listado de fuentes a consultar por parte del recopilador, pudiendo aumentar el volumen de artículos diarios y dando mucha flexibilidad de

cara al futuro ya que de este modo se podrán incluir categorías emergentes al modelo clasificador.

A continuación, se detallará el tratamiento dado a los datos y su uso para el entrenamiento del clasificador.

2.2 Desarrollo del clasificador

En el siguiente apartado se explicará el desarrollo realizado para lograr el mejor clasificador posible, comenzando con una pequeña introducción al mundo de la inteligencia artificial, siguiendo con la explicación del proceso seguido y finalizando con unas pequeñas conclusiones.

2.2.1 Introducción

Una de las cualidades diferenciadoras de Tentu es que dispone de una gran variedad de contenido recogida de distintas fuentes y distribuida sobre muchas categorías, para así poder ofrecer al usuario final una experiencia de uso única, visualizando el contenido relevante a sus temáticas favoritas.

El contenido puede ser introducido en Tentu de dos maneras, mediante un creador de contenidos o una fuente RSS. En el caso de un creador de contenidos, al crear el articulo o evento, este debe seleccionar al menos una categoría o pueblo al que tenga relación, y como lo ha establecido el propio creador del contenido, no debería de haber ningún tipo de fallo en cuanto a que el contenido se introduzca en la categoría incorrecta, ya que no depende de ningún sistema externo.

En cambio, la obtención de datos mediante fuentes RSS necesita una forma automática para realizar la categorización del contenido. En algunas ocasiones, las fuentes RSS recogen artículos y eventos relacionadas con una sola categoría. Por ejemplo, en el caso de un periódico que disponga de una fuente RSS distinta por cada tema o apartado del periódico, como pueden ser política, economía, deportes… Sin embargo, en la mayoría de las fuentes de información RSS, el nivel de especificación del contenido es mucho menor y más generalista, sin especificar una categoría e incluyendo noticias de distintas temáticas en una misma fuente RSS.

Para solventar esta situación, el centro de terminología y lexicografía UZEI, junto con la colaboración de ISEA puso en marcha un sistema de categorización de contenidos, que actualmente está disponible para su uso pero que sigue en constante desarrollo.

Tras realizar varias pruebas, el equipo observo que el contenido que estaba siendo introducido a Tentu no estaba siendo categorizado correctamente, consiguiendo un porcentaje de acierto menor al 75%, lo que conlleva que la experiencia de uso y la granularidad que se pretende ofrecer no se cumpla.

Con el objetivo de obtener un nivel de granularidad mayor en cuanto al clasificado de contenidos, el equipo ha decidido diseñar y desarrollar un sistema que sustituya o complemente el servicio ofrecido por UZEI.

En cuanto al diseño del sistema a desarrollar, se realizará un modelo para predecir la/las categorías a las que pertenece un contenido en específico, utilizando técnicas de clasificación supervisada (aprendizaje supervisado) en el campo de aprendizaje automático. Antes de comenzar resulta interesante hacer una aclaración sobre el mundo de la inteligencia artificial.

2.2.1.1 Introducción al análisis de datos

Hoy en día el análisis de datos alberga muchas variantes en su interior, como pueden ser el aprendizaje automático (Machine Learning [15]) o el aprendizaje profundo (Deep Learning [16]), entre otros. Antes de profundizar en las implementaciones de cada uno, resulta interesante aclarar las diferencias entre cada uno de ellos.

Cabe destacar que la mayoría de IA existentes son realmente ordenadores diseñados para resolver problemas, o más bien para hacer predicciones basándose en los datos que tiene, como el cerebro humano [17]. Normalmente, se introducen un gran número de datos (como los números del 1 al 10), posteriormente se hace un modelo (X + 1 empezando en el 0) y la inteligencia artificial realiza una predicción en base a ese modelo.

Lo que diferencia a las aplicaciones que usan Inteligencia Artificial de los que no es que no hay que programarla específicamente para cada escenario [18]. Es decir, el objetivo es o bien realizar un análisis e introducir un gran número de datos válidos para el entrenamiento (en el caso del Machine Learning, aprendizaje automático), o introducir los datos sin análisis previo para que aprenda y valore por sí mismo los datos más importantes para determinar la clase (Deep Learning). Si bien existen múltiples variantes de cada uno, se pueden definir de manera general de la siguiente forma [19]:

- IA (Inteligencia Artificial): Un sistema que es capaz de imitar el razonamiento humano.
- ML (Machine Learning): Un subconjunto de Inteligencia Artificial donde las personas «entrenan» a las máquinas para reconocer patrones basados en los datos introducidos y hacer sus predicciones.
- DL (Deep Learning): Un subconjunto de ML en el que la máquina es capaz de razonar y sacar sus propias conclusiones, aprendiendo por sí misma, sin que el desarrollador tenga que intervenir de ninguna forma.

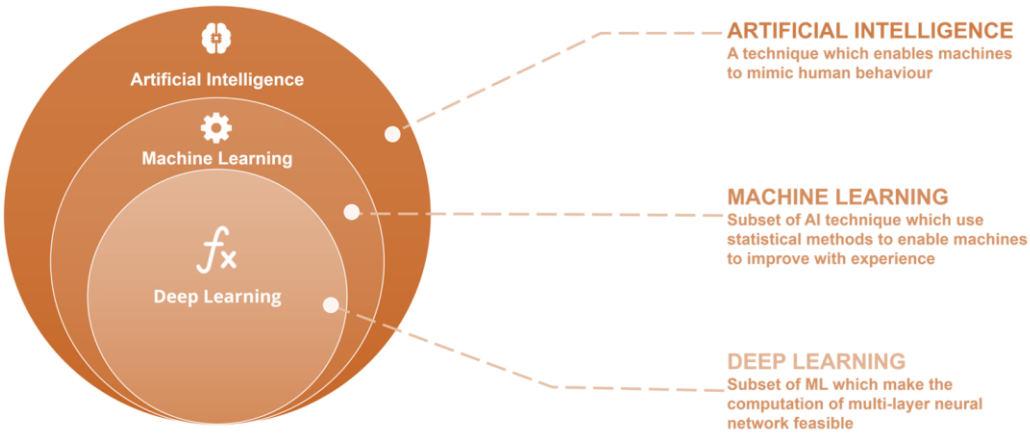


Ilustración 14: Análisis de la inteligencia artificial

2.2.1.1.1 Inteligencia artificial

La definición más amplia posible de lo que es la Inteligencia Artificial es simplemente que se trata de una máquina o equipo capaz de pensar como un ser humano [19]. Puede ser tan simple como seguir un diagrama de flujo lógico, o puede ser un ordenador casi humano que es capaz de aprender una gran cantidad de entradas sensoriales y aplicar ese conocimiento a nuevas situaciones. Esta última parte es la clave, de hecho: la IA ideal, por decirlo de alguna forma, es aquella que puede conectar todos los datos aprendidos para tener la capacidad de manejar cualquier situación.

Es ideal porque la inteligencia artificial está bastante lejos de eso, por ejemplo, Alexa, de Amazon, puede ser un buen asistente, pero no puede pasar la famosa prueba de Turing [20].

2.2.1.1.2 Aprendizaje automático

Como se ha comentado en el apartado anterior, el aprendizaje automático es la rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que permitan a las máquinas aprender [15]. De forma más concreta, se trata de crear algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de una información suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento, es decir, un método que permite obtener por generalización un significado general a partir de casos particulares.

Cuando se han observado todos los casos particulares la generalización se considera completa, por lo que la generalización a la que da lugar se considera válida. No obstante, en la mayoría de los casos es imposible obtener una generalización completa, por lo que el enunciado al que da lugar queda sometido a un cierto grado de incertidumbre, y por lo tanto no se puede considerar valido como tal.

A un nivel muy básico, se podría decir que una de las tareas del aprendizaje automático es intentar extraer conocimiento sobre algunas propiedades no observadas de un objeto basándose en las propiedades que sí que han sido observadas. [21]En otras palabras, predecir comportamiento futuro a partir de lo que ha ocurrido en el pasado. Un ejemplo de mucha

actualidad sería, el de predecir si una determinada canción le va a gustar a un oyente basándose en las valoraciones que ese mismo oyente ha hecho de otras canciones que sí ha escuchado.

En cualquier caso, como la inteligencia artificial está relacionada con el aprendizaje, se debe definir qué se entiende por aprendizaje. El aprendizaje puede tener diversos significados dependiendo del contexto, en el informático se podría decir que se refiere a aquello que la máquina pueda aprender a partir de la experiencia, no a partir del reconocimiento de patrones programados a priori [22]. Por tanto, consiste en alimentar la experiencia de la máquina por medio casos/ejemplos con los que entrenarse para, posteriormente, aplicar los patrones que haya reconocido sobre objetos que no haya visto previamente.

Hay un gran número de problemas afrontables a través del machine learning, la principal diferencia entre ellos está en el tipo de objetivo que intentan predecir. Algunas de las habituales son [23]:

- Regresión: Intentan predecir un valor real. Por ejemplo, predecir el valor de la bolsa mañana a partir del comportamiento de la bolsa que está almacenado (pasado). O predecir la nota de un alumno en el examen final basándose en las notas obtenidas en las diversas tareas realizadas durante el curso.
- Clasificación (binaria o multiclase): Intentan predecir la clasificación de objetos sobre un conjunto de clases prefijadas. Por ejemplo, como es el objetivo de este apartado, clasificar si una determinada noticia es de deportes, entretenimiento, política, etc. Si solo se permiten 2 posibles clases, entonces se llama clasificación binaria; si se permiten más de 2 clases, estamos hablando de clasificación multiclase.

Normalmente, cuando se aborda un nuevo problema de machine learning, lo primero que se debe hacer es enmarcarlo dentro de alguna de las clases anteriores, tal y como se hará en el apartado 2.2.2, ya que dependiendo de cómo se clasifique será la forma en que se pueda medir el error cometido entre la predicción y la realidad.

Por otra parte, dependiendo del tipo de salida que se produzca y de cómo se aborde el tratamiento de los ejemplos, los diferentes algoritmos de AA se pueden agrupar en:

- Aprendizaje supervisado [24]: Se genera una función que establece una relación entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada ejemplos de los que sabemos su clasificación correcta. Un ejemplo de este tipo de algoritmo es el problema de clasificación de textos mencionado anteriormente.
- Aprendizaje no supervisado [25] [26]: Donde el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer su clasificación correcta. Por lo que se busca que el sistema sea capaz de reconocer patrones para poder etiquetar las nuevas entradas.

En el caso de este proyecto se van a tratar los dos tipos de aprendizajes con el fin de analizar cuál es el que mejor resultado puede dar.

2.2.1.1.3 Aprendizaje profundo

Normalmente, las personas aprenden de la experiencia. Cuanto más variadas sean las vivencias, más aprenden. En la disciplina de la inteligencia artificial que se conoce como aprendizaje profundo, esto mismo se puede decir de las máquinas potenciadas por el software y el hardware de inteligencia artificial. Las experiencias a través de las cuales aprenden las máquinas se definen mediante los datos que adquieren, y la cantidad y la calidad de estos datos determinan cuánto pueden aprender [27].

El aprendizaje profundo se sitúa en el interior del aprendizaje automático. A diferencia de los algoritmos tradicionales de aprendizaje automático, muchos de los cuales tienen una capacidad finita de aprendizaje independientemente de cuántos datos adquieran, los sistemas de aprendizaje profundo pueden mejorar su rendimiento al poder acceder a un mayor número de datos, o lo que es lo mismo, hacer que la máquina tenga más experiencia.

Las redes neuronales del aprendizaje profundo aprenden mediante la detección de estructuras complejas de los datos que reciben. Al crear modelos computacionales compuestos por varias capas de procesamiento, las redes pueden crear uno o varios niveles de abstracción que representen los datos.

Por ejemplo, un modelo de aprendizaje profundo conocido como redes neuronales convolucionales, del cual se habla en el apartado 2.2.3.3.3, se puede entrenar con un gran número (de millones) de imágenes, por ejemplo, las que contienen gatos. Este tipo de red neuronal normalmente aprende de los píxeles que contienen las imágenes que adquiere. Puede clasificar grupos de píxeles que representan las características de un gato, con grupos de características como las garras, las orejas y los ojos, lo que indicaría la presencia de un gato en la imagen [27].

Ilustración 15: Imágenes de gatos utilizadas para entrenar una red neuronal convolucional

Con el aprendizaje profundo, todo lo que necesita es ofrecer al sistema un gran número de imágenes de gatos, tras lo cual el sistema aprende de forma autónoma las características que representan un gato.

En muchas tareas, como la visión por ordenador, el reconocimiento de voz, la traducción automática y la robótica, el rendimiento de los sistemas de aprendizaje profundo supera

enormemente el de los sistemas de aprendizaje automático convencional. Esto no significa que crear sistemas de aprendizaje profundo sea relativamente fácil en comparación con los sistemas de aprendizaje automático convencional. Si bien el reconocimiento de características es autónomo en el aprendizaje profundo, hay que ajustar miles de hiperparámetros para que el modelo de aprendizaje profundo sea realmente efectivo [28].

Tras introducir el mundo del análisis de datos y sus ramificaciones, es hora de definir el problema a afrontar.

2.2.2 Problema a afrontar

En cuanto al clasificador temático, como el propio nombre indica se trata de un problema de clasificación. En este caso, el modelo debe de ser capaz de leer un artículo y responder con un conjunto de probabilidades asignada a cada clase, de modo que la más alta sea la clase o categoría asignada.

Para ejemplo, la siguiente ilustración:

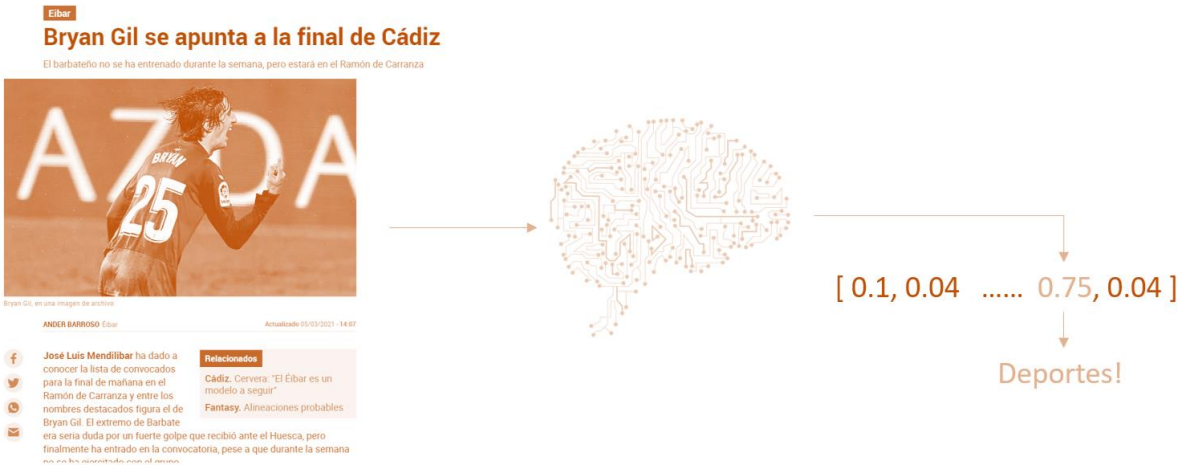


Ilustración 16: Ejemplo del problema que se quiere afrontar

El problema se va a afrontar mediante aplicaciones tanto de técnicas supervisadas (aprendizaje automático con SciKit Learn y aprendizaje profundo con Keras), como de no-supervisadas como puede ser el clustering mediante K-means.

Para ello, en el siguiente apartado se detallarán algunos de los algoritmos más relevantes, así como el tratamiento base realizado a los datos recopilados mediante el software desarrollado en el apartado anterior.

2.2.3 Implementación

En este apartado se detallarán algunos de los algoritmos utilizados, así como el tratamiento realizado a los datos. Dado que gran parte de los algoritmos hacen uso del mismo tratamiento, se comenzará con ello. En caso de que alguno de los algoritmos tenga un

tratamiento adicional de datos, como pueden ser las neuronas LSTM del aprendizaje profundo, se indicará en su propia explicación.

2.2.3.1 Información y tratamiento de los datos

Por un lado, en cuanto a los datos, resulta interesante destacar que todas las pruebas y resultados mostrados en este documento están realizados con un conjunto de datos que contiene artículos sobre las siguientes 20 temáticas:

- Comunicación y publicidad
- Arte
- Motor
- Cine
- Cultura
- Decoración
- Economía
- Moda
- Alimentación, cocina y gastronomía
- Medicina
- Música
- Naturaleza, medio ambiente y ecología
- Opiniones
- Fotografía
- Política y sindicatos
- Ciencia y tecnología
- Sociedad
- Deportes
- Turismo y viajes
- Transporte

Además, con el fin de que no haya un desequilibrio en el dataset, se ha desarrollado un software adicional llamado equilibrador. La labor de este software consiste en asegurar que cada una de las temáticas contiene la misma cantidad de artículos únicos en su interior y que la longitud de estos artículos cumple un mínimo. Asimismo, realiza una pequeña limpieza de pequeñas proposiciones recogidas accidentalmente debido a la dificultad de definir un código XPATH exacto.

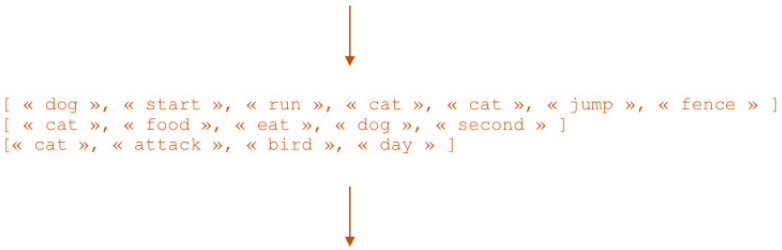
Por otro lado, además del tratamiento realizado por el recopilador (explicado en el apartado 2.1.4 de este documento) se ha realizado un tratamiento adicional a los datos, en función de que requería cada uno de los algoritmos.

Este tratamiento adicional, principalmente consiste en la tokenización de cada uno de los artículos que compone el dataset [29]. Para ello, se ha hecho uso de la librería SciKit Learn que trata de asignar un identificador entero fijo a cada palabra que aparezca en cualquier documento del conjunto de entrenamiento (construyendo un diccionario de palabras con índices enteros).

Tras realizar esta identificación, por cada documento o artículo se cuenta el número de apariciones de cada palabra y se almacena. El principal problema o cuello de botella de esta técnica es la cantidad de memoria RAM que consume en el sistema, ya que dependiendo del número de palabras únicas que tenga nuestra base de textos, este tomará más o menos espacio.

Se adjunta la siguiente ilustración con el fin de comprender mejor lo explicado hasta este punto:

« The dog started to run after the cat, but the cat jumped over the fence. »
« The cat's food was eaten by the dog in a few seconds. »
« The cat attacked the bird the other day. »



Bag-Of-Words representation

	Dog	Start	Run	Cat	Jump	Fence	Food	Eat	Second	attack	Bird	Day
1	1	1	1	**2**	1	1	0	0	0	0	0	0
2	1	0	0	1	0	0	1	1	1	0	0	0
3	0	0	0	1	0	0	0	0	0	1	1	1

Ilustración 17: Resumen sobre parte del preproceso realizado a los datos

La guía de la librería [30] recomienda pasar de ocurrencias de palabras en cada documento, a frecuencia de aparición de ellas, aunque también menciona que no siempre da buenos resultados, por lo que se harán pruebas tanto con apariciones como con frecuencias.

El principal motivo para recomendar las frecuencias en lugar de las apariciones es que los documentos más largos tendrán valores de recuento más altos que los cortos, aunque ambos hablen de la misma categoría.

Para evitar posibles discrepancias, divide el número de apariciones de cada palabra en un documento por el número total de palabras de este. Además, haciendo esto se consigue reducir las ponderaciones de las palabras que aparecen en muchos documentos del corpus, por lo que son menos informativas que las que solo aparecen en una parte más pequeña del corpus [31].

Técnicamente a este proceso se le denomina Tf-Idf que significa "Term Frecuency times Inverse Document Frequency".

Tras analizar y explicar el tratamiento adicional realizado a los datos, se explicarán algunos de los algoritmos utilizados en las pruebas, así como el entorno utilizado para el entrenamiento

2.2.3.2 Análisis de algoritmos de aprendizaje automático

Como se ha detallado en la introducción a este apartado, el aprendizaje automático se basa en intentar extraer conocimiento sobre algunas propiedades no observadas de un objeto basándose en las propiedades que sí que han sido observadas.

Para ello hacen falta diferentes algoritmos que se encarguen de generar un modelo que haya extraído ese conocimiento de los datos de entrenamiento. En este apartado se detallarán algunos de los algoritmos utilizados en las pruebas.

Antes de empezar, resulta interesante destacar que el criterio seguido para seleccionar unos algoritmos frente a otros es la aparición de los algoritmos en diferentes guías, como pueden ser las de la propia librería o incluso consejos/opiniones de diferentes expertos.

2.2.3.2.1 Support Vector Machine

Dado un conjunto de ejemplos de entrenamiento, una SVM los representa como puntos en el espacio, separando las clases por la mayor distancia posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, según su proximidad serán asignadas a una u otra clase. Es decir, si un punto nuevo pertenece a una categoría o la otra [32].

En la ilustración, se puede observar cómo H1 no separa las clases, en cambio H2 si lo hace, aunque no con la separación máxima. H3 sería el modelo óptimo.

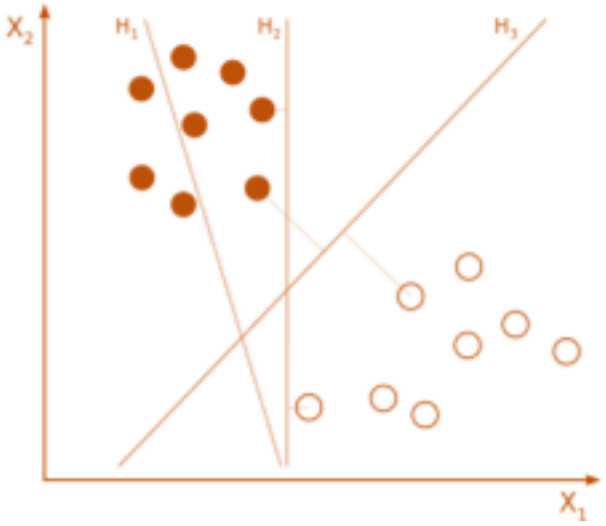


Ilustración 18: Ejemplo gráfico de Support Vector Machine

2.2.3.2.2 (Multinomial) Naive Bayes

Es un clasificador probabilístico basado en el teorema de Bayes y algunas hipótesis simplificadoras. Asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica, dada la clase variable. Como ventaja, cuenta con que necesita una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios para la clasificación.

El clasificador multinomial de Naive Bayes es adecuado para la clasificación con características discretas (por ejemplo, recuentos de palabras para la clasificación de textos, como el tokenizador comentado previamente). La distribución multinomial normalmente requiere recuentos enteros de características. Sin embargo, en la práctica, los recuentos

fraccionarios, como tf-idf, también pueden funcionar, lo que es un gran punto a favor de este algoritmo. [33]

2.2.3.2.3 Logistic Regression

Logistic Regression es un algoritmo de predicción que utiliza variables independientes para predecir la variable dependiente, al igual que la Regresión Lineal, pero con la diferencia de que la variable dependiente debe ser categórica. Las variables independientes, en cambio, pueden ser numéricas o categóricas.

La principal diferencia con la regresión lineal es que, en lugar de trazar una recta, es posible trazar una curva, con el fin de adaptarse mejor a los datos. [34]

A modo de ejemplo, en la siguiente ilustración se puede observar que el eje Y proporciona valores de 0 a 1. Esto supone que al calcular la función sigmoidea de X se obtendrá una probabilidad (entre 0 y 1, obviamente) de que una observación, en nuestro caso un documento, pertenezca a una clase u otra.

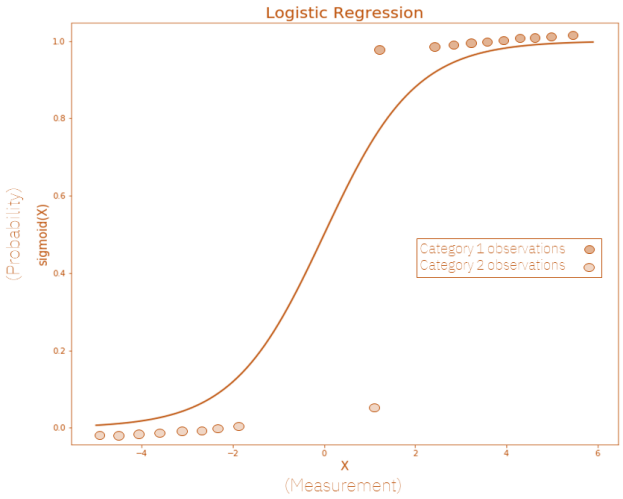


Ilustración 19: Ejemplo gráfico de Logistic Regression

2.2.3.2.4 Decision Tree Classifier

Decision Tree Classifier hace uso de un árbol de decisión como modelo predictivo que mapea observaciones sobre un artículo a conclusiones sobre el valor objetivo del artículo. Las hojas representan etiquetas de clase y las ramas las características que conducen a esa clase. Cada nodo representa uno de los valores de entrada y sus nodos hijos representan un posible valor para esa variable de entrada. Las hojas representan un posible valor para la entrada dada. [35]

Tras analizar los principales algoritmos de aprendizaje automático, toca hablar sobre los de aprendizaje profundo.

2.2.3.3 Análisis de algoritmos de aprendizaje profundo

Para implementar algoritmos de aprendizaje profundo existen diferentes herramientas, en este caso el desarrollo se ha centrado en Keras y los algoritmos o tipos de redes neuronales aplicados han sido ANNs, CNNs y LSTMs. En el siguiente apartado se analizarán cada una de las opciones.

2.2.3.3.1 Artificial Neural Network

Las Artificial Neural Networks consisten en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos u otros valores de salida, dependiendo del tipo de neurona [36].

Cada neurona está conectada con otras a través de unos enlaces. En estos enlaces el valor de salida de la neurona anterior es ponderado por un peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo, a la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que no se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

Como se ha mencionado, estos sistemas aprenden por si mismos por lo que sobresalen en áreas donde la detección de características es difícil de expresar con la programación convencional, como puede ser el caso que se analiza a través de este proyecto, la clasificación de textos multiclase. Para realizar este aprendizaje automático, normalmente, se intenta minimizar una función de pérdida que evalúa la red en su total. Los valores de los pesos de las neuronas se van actualizando, con el fin de reducir el valor de la función de pérdida. Este proceso se realiza mediante la propagación hacia atrás o backpropagation [37].

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más abstractas. Las redes neuronales actuales suelen contener desde unos miles a unos pocos millones de unidades neuronales.

En este caso se ha trabajado en una red neuronal simple, en la que la entrada son los documentos tokenizados, y se introduce en una sola capa con activación 'ReLu', teniendo por salida una capa densa con la misma cantidad de neuronas que temáticas el dataset contiene, en este caso, como se ha indicado en el 2.2.3.1, 20.

Por otro lado, se ha añadido una capa de Dropout con el fin de prevenir overfitting, es decir, que el modelo se ajuste demasiado al conjunto de entrenamiento. Técnicamente, la capa Dropout establece aleatoriamente las unidades de entrada a 0 con una frecuencia constante en cada paso durante el tiempo de entrenamiento, lo que ayuda a prevenir el overfitting [38].

Cabe destacar que una gran ventaja de esta red es que no es necesario un tratamiento adicional de los datos y con el básico es suficiente.

En el apartado 2.2.4 se mostrarán los resultados conseguidos mediante este desarrollo. Ahora, toca hablar sobre las redes neuronales con Word Embeddings.

2.2.3.3.2 ANN con Word Embeddings

El texto se considera una forma de datos secuenciales similar a los datos de series temporales que se tienen en los datos meteorológicos o financieros. En los modelos anteriores, tanto de aprendizaje profundo como automático, se ha representado una secuencia completa de palabras como un único vector de características.

En los Word Embeddings las palabras o frases del lenguaje natural son representadas como vectores de números reales que pueden ir modificándose durante el entrenameinto. Esto significa que los Word Embeddings, recogen más información en menos dimensiones [39].

Se debe tener en cuenta que las Word Embeddings no comprenden el texto como lo haría un ser humano, sino que mapean la estructura estadística del lenguaje utilizado en el corpus. Su objetivo es mapear el significado semántico en un espacio geométrico llamado *embedding space* [40].

Si estas relaciones se realizan de forma adecuada, las palabras semánticamente similares se acercarían en el *embedding space*, como podría ser el caso de los colores o números. En caso de funcionar correctamente, incluso se podrían realizar operaciones de aritmética vectorial. Un ejemplo famoso en este campo de estudio es la capacidad de mapear Rey – Hombre + Mujer = Reina.

Existen dos opciones para conseguir este Word Embedding. Una es crearla durante el entrenamiento de la red neuronal, partiendo de 0 y otra es utilizar preentrenadas por terceros (gratuito de todas formas) [41]. En este segundo caso, se puede elegir dejarla sin modificar o que sea la propia red neuronal quien vaya modificando la agrupación según le convenga.

Tras detallar la base del funcionamiento de las redes neuronales convolucionales, es conveniente concretar el tratamiento adicional realizado a los datos, además del explicado en el apartado 2.2.3.1.

El principal problema de los datos del dataset, es que cada secuencia de texto tiene una longitud de palabras variable. Para contrarrestar esto, se utiliza una función que la misma herramienta Keras proporciona. Esta función simplemente rellena la secuencia de palabras con ceros hasta un límite establecido, en caso de que supere el límite, lo recorta, dejando así todas las secuencias con la misma longitud [42].

Para crear una Word Embedding propia hace falta una capa de embedding a la red neuronal que la misma Keras proporciona. A esta hay que proporcionarle diferentes parámetros como son el tamaño del vocabulario (índice prefijado en el tokenizador), el tamaño que debe tener el vector de la Word embedding y por último la longitud que tienen cada una de las secuencias que forman el dataset.

Tras la capa de embedding, existen múltiples opciones para configurar el resto de la red neuronal. Se han analizado varias, tanto generando una embedding propia como de terceros y su resultado puede observarse en el apartado 2.2.4.

2.2.3.3.3 Convolutional Neural Network

En el apartado anterior se ha visto cómo funcionan las redes neuronales corrientes y las que incluyen Word Embeddings. Resumiendo, se puede destacar que las neuronas se alimentan de entradas que las neuronas ponderan pasando por una función de activación y finalmente dar salida a la siguiente neurona.

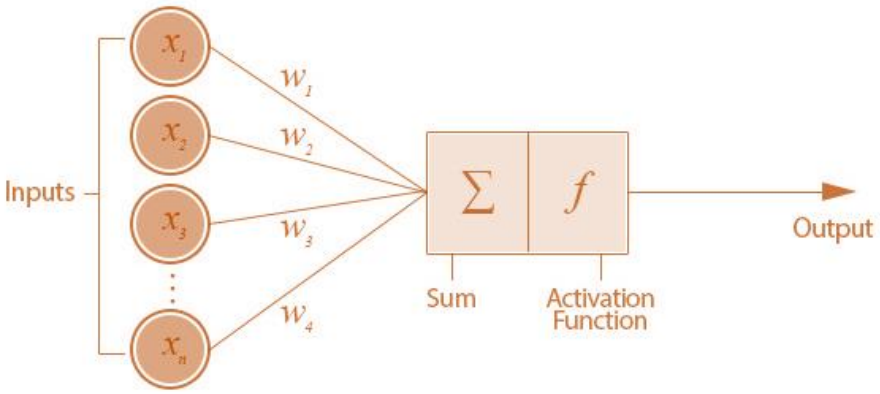


Ilustración 20: Resumen gráfico de cómo funcionan las redes neuronales convencionales

En cambio, una red convolucional trabaja con un volumen concreto de entradas. La principal diferencia con las redes neuronales convencionales es que cada capa trata de encontrar patrones de datos más complejos. Debido a ello, uno de sus principales usos hoy en día son detectar patrones en visión artificial o la detección de anomalías. [42]

Cuando se trabaja con datos secuenciales, como el texto, se trabaja con convoluciones unidimensionales, pero la idea y la aplicación siguen siendo las mismas que con las multidimensionales, captar los patrones de la secuencia, que a su vez se vuelven más complejos con cada capa convolucional añadida. [43]

En la siguiente ilustración se puede ver cómo funciona una convolución de este tipo. Comienza tomando un fragmento de la entrada del tamaño del núcleo del filtro. Con este fragmento se toman las características clave y se comprimen en el espacio latente.

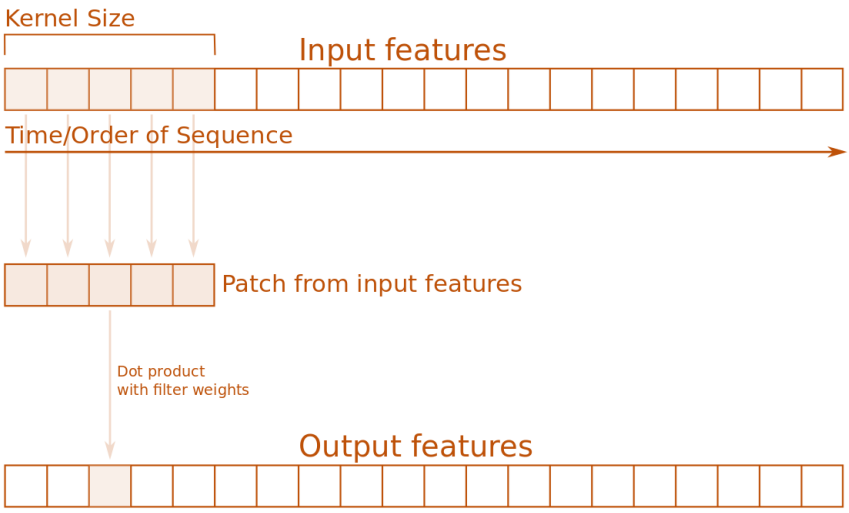


Ilustración 21: Ejemplo gráfico de cómo funciona la convolución en el tratamiento

(captura sacada del libro (Deep Learning with Python [44])

La red neuronal unidimensional es invariante a las traslaciones o movimientos, lo que significa que ciertas secuencias o patrones pueden ser reconocidas en una posición diferente. Trasladando esto a un ejemplo práctico, se podría decir que al modelo le daría igual que la palabra 'Messi' apareciese al comienzo o al final del documento, ya que aparezca donde aparezca lo relacionaría con el deporte, lo cual puede ser muy útil a la hora de reconocer patrones en el texto.

Para terminar, es necesario indicar que la implementación de esta red neuronal ha requerido un pequeño tratamiento para que todos los documentos cuenten con la misma longitud. Para ello se ha utilizado la misma función que en los ANN con Word Embeddings.

2.2.3.3.4 Long Short-Term Memory

El último tipo de red neuronal por detallar antes de observar los resultados son los Recurrent Neural Networks, que tiene en su interior los Long Short-Term Memory o LSTMs a partir de ahora, que son los que se han implementado. Por poner un ejemplo entendible por todos, los seres humanos no empiezan a pensar desde cero cada segundo. Cuando se trata de leer este documento, por ejemplo, se entiende cada palabra en base a la comprensión de las palabras anteriores. No se empieza a pensar desde cero otra vez. Los pensamientos, por decir de alguna forma, tienen persistencia [45].

Las redes neuronales tradicionales no pueden hacer esto, y parece un gran defecto. Por ejemplo, imagina que se quiere clasificar qué tipo de evento está ocurriendo en cada momento de una película. No está claro cómo una red neuronal tradicional podría utilizar su razonamiento sobre los acontecimientos anteriores de la película para informar sobre los posteriores.

Las redes neuronales recurrentes abordan este problema. Son redes con bucles que permiten que la información persista.

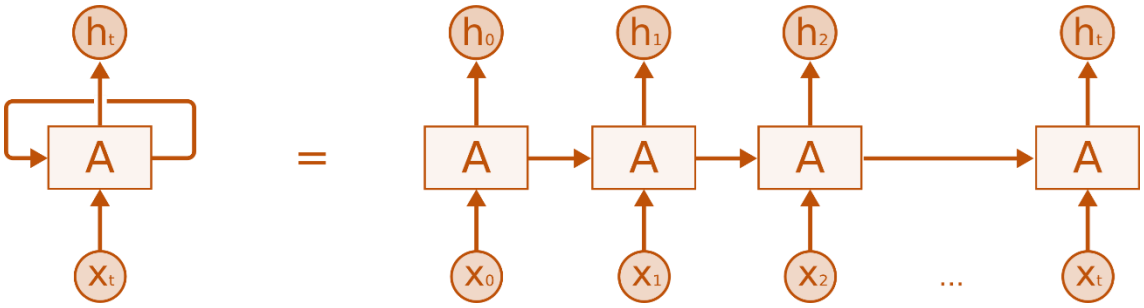


Ilustración 22: Desglose de una neurona recurrente

A veces solo se necesita información reciente o a corto plazo para realizar una predicción. Como por ejemplo un modelo que predice la siguiente palabra basándose en las anteriores. Poniendo el ejemplo de intentar predecir la última palabra de “las nubes están en el cielo”, no se necesita más contexto, es evidente que la siguiente palabra será cielo.

Aunque hay veces que se necesita contexto más tardío o a largo plazo. Intentando predecir la última palabra de “Me crie en Francia… por lo tanto hablo francés con fluidez”, la información a corto plazo sugiere el nombre del idioma, pero debe recurrir a la información a largo plazo, en este caso Francia para saber que la siguiente palabra es francés.

El principal problema que contienen estas RNNs principalmente es que no son capaces de manejar memoria a largo plazo. Aunque a su variante, las LSTMs no tienen ese problema.

Las LSTMs están diseñadas para evitar el problema de la dependencia a largo plazo. Para conseguirlo, en lugar de utilizar una capa tanh para unir los bucles de la neurona, utilizan cuatro componentes [46]:

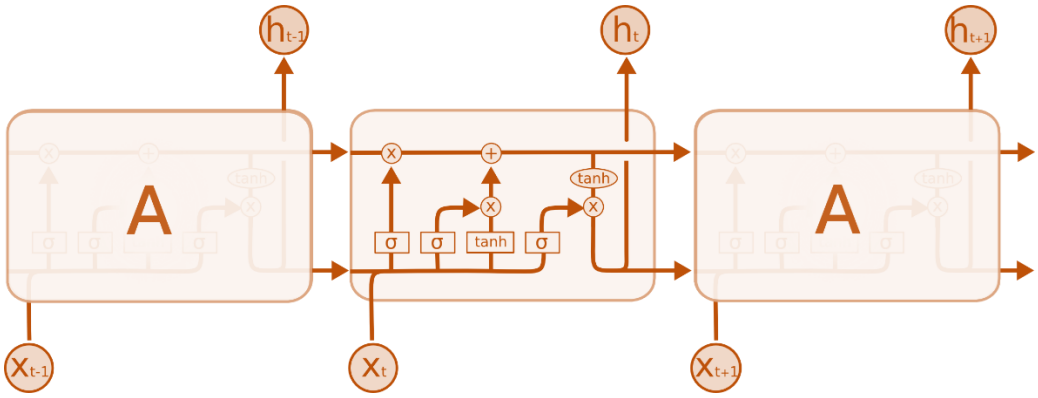


Ilustración 23: Profundización sobre el funcionamiento de una neurona LSTM

Los componentes de la unión son las siguientes:

Ilustración 24: Componentes de unión de una neurona LSTM

El funcionamiento básico se basa en que a través de la línea superior horizontal fluye la información sin apenas alteraciones lineales. Es en la parte inferior donde la neurona trabaja con puertas lógicas para eliminar o añadir información al estado de la célula. Son las puertas las que valoran que información es relevante o no, en este caso está compuesta por una capa de red neuronal sigmoide, dando como resultado un valor entre 0 y 1, lo que describe la cantidad de cada componente que debe dejar pasar.

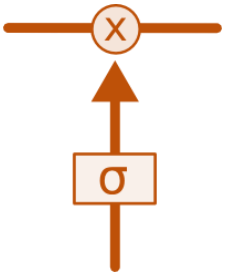


Ilustración 25: Ejemplo de puerta lógica en el interior de la neurona LSTM

Tras aclarar esto, la traza de una célula LSTM es la siguiente:

El primer paso es decidir qué información se debe desechar del estado o memoria de la célula. Esta decisión la toma la “forget gate” que corresponde a la primera capa sigmoide que se sitúa en la parte inferior izquierda de la ilustración 20.

En la siguiente etapa se decide qué información nueva se va a tener en cuenta para almacenar en el estado de la célula. Esto se consigue con dos capas, una sigmoide que decide que valores van a actualizar su valor y otra tanh que crea un vector de valores candidatos a ser añadidos. Todo este proceso corresponde a la parte central inferior de la ilustración previa.

A continuación, se actualiza el estado de la célula teniendo en cuenta el trabajo realizado en los pasos anteriores. Para finalizar se decide qué salida se debe dar a la siguiente célula, para ello se transmitirá una versión filtrada mediante una capa sigmoide del estado actual de la célula. Tras realizar ese filtro se pasa por una capa de tanh, de modo que solo salgan las partes que se han decidido [46].

Se debe de tener en cuenta que todo este proceso se realiza por cada neurona LSTM de la red que se implemente, lo que supone un gran número de variables (pesos y bias de cada uno de los procesos) a optimizar por cada neurona. Son un tipo de neurona muy potentes, pero en su contra juegan que el tiempo de aprendizaje es alto y se deben diseñar cuidadosamente.

Por último, en cuanto al tratamiento específico de los datos para la implementación de los datos, se debe destacar que el único requisito de este tipo de neuronas es que todas las secuencias cuenten con la misma longitud. Es por ello por lo que se ha hecho uso de la misma funcionalidad que en las CNN, es decir recortar o alargar cada una de las secuencias.

Tras analizar los algoritmos de aprendizaje profundo, es hora de analizar los de aprendizaje no supervisado.

2.2.3.4 Análisis de algoritmos de aprendizaje no-supervisado

Como se ha mencionado en la introducción a la sección, el aprendizaje no-supervisado se diferencia principalmente en que el algoritmo no conoce la clasificación correcta, por lo que busca que el sistema sea capaz de reconocer patrones para poder etiquetar las nuevas entradas.

Existen diversos algoritmos de aprendizaje no-supervisado, siendo los más famosos K-Means, DBSCAN o MeanShift [47]. Entre ellos el más sencillo de implementar es K-Means, en la siguiente sección se detallará su funcionamiento.

2.2.3.4.1 K-Means

K-means tiene como objetivo dividir los datos en k clusters de forma que los puntos de datos del mismo cluster sean similares y los puntos de datos de los diferentes clusters estén más alejados. La similitud de dos puntos se determina por la distancia entre ellos [48].

Hay muchos métodos para medir la distancia, la euclidea es la utilizada por defecto por el algoritmo.

Las principales ventajas de K-means es que es fácil de entender e implementar, además de que maneja estupendamente datasets largos.

Por otro lado, sus debilidades están relacionadas con la sensibilidad al número de clusters/centroides elegidos. Incluso después de utilizar técnicas como Elbow [14]es difícil generar buenos clusters. Aunque este método en este caso no es útil ya que interesa que las 20 categorías estén bien diferenciadas y que el valor de Silhouette[15] se aleje de 0 lo máximo posible.

Otra de las desventajas principales es que no funciona correctamente cuando el dataset contiene valores atípicos, ya que de esta forma los centroides de los clusters pueden ser arrastrados por ellos, lo que da lugar a clusters sesgados.

Además, puede resultar lento cuando el número de clusters a definir es alto.

Tras detallar el funcionamiento de K-means, se analizará una alternativa, DBSCAN.

2.2.3.4.2 DBSCAN

DBSCAN al contrario de K-Means se centra en la densidad a la hora de crear los clusters, ignorando las áreas con poca densidad, considerándolas ruido o valores atípicos [49] [50].

DBSCAN cuenta con la ventaja de que, al contrario de K-Means, funciona bien con datasets con muchos valores atípicos, ya que los ignora. Por otro lado, funciona medianamente mal con

[14] Técnica que ayuda a encontrar el óptimo número de clusters (K) para la correcta clasificación del modelo.
[15] Método de validación de los modelos de clustering. Valora si las agrupaciones o clusters están diferenciados los unos de los otros, proporcionando un valor entre -1 y +1.

datasets con densidad variable y es complicado buscar un valor Epsilon[16] que se adapte a la cantidad de clusters que se desee. Asimismo, no se puede fijar una cantidad de clusters a modelar como si se podía en K-Means. En cambio, sí que se puede fijar el número mínimo de puntos que debe tener el cluster para considerarse tal.

Teniendo en cuenta lo dicho, es un algoritmo complicado de implementar correctamente si se quiere sacar un buen resultado. Además de que tanto DBSCAN como K-Means no tienen un método para validarse realmente como si lo tienen los algoritmos de aprendizaje supervisado. Esto es algo de lo que se hablará en la sección de resultados.

2.2.3.5 Entorno utilizado en el entrenamiento

Como se puede observar en el pliego de condiciones del apartado 1.6, el equipo con el que se está desarrollando el proyecto no es lo suficientemente potente como para ejecutar los algoritmos detallados en los apartados anteriores.

Por este motivo se ha decidido ejecutar todo el código en un entorno en la nube como es Google Colab. Colab es un servicio cloud, basado en los notebooks de Jupyter (ficheros de formato ipynb), que permite el uso gratuito de las GPUs y TPUs de Google, con librerías como Scikit-learn, PyTorch, TensorFlow, Keras y OpenCV, todo ello bajo Python [51]. Además, para el almacenamiento de datasets, es realmente sencillo complementarlo con Google Drive, que tiene almacenamiento más que suficiente para ello.

2.2.4 Resultados obtenidos

Para comprobar los resultados obtenidos se debe validar como de buenos son los modelos generados. Antes de ello resulta interesante aclarar los siguientes conceptos referentes a la división del dataset para la validación [52].

- Conjunto de entrenamiento: El conjunto de datos real que se utiliza para entrenar el modelo (pesos y sesgos en el caso de una red neuronal). El modelo ve y aprende de estos datos.
- Conjunto de validación: El conjunto de validación se utiliza para evaluar un modelo determinado, esta una evaluación frecuente. Se utiliza para el ajuste de hiperparámetros.
- Conjunto de test: La muestra de datos se utiliza para proporcionar una evaluación imparcial del ajuste del modelo final en el conjunto de datos de entrenamiento.

En la siguiente ilustración se puede ver lo explicado de una forma gráfica:

[16] Se define como la distancia máxima entre dos puntos que deben ser considerados como puntos vecinos (pertenecientes al mismo clúster).

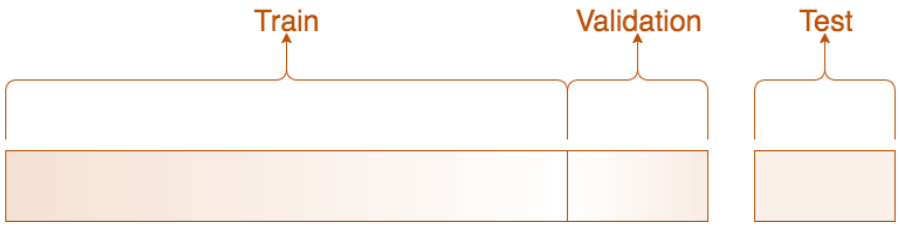


Ilustración 26: Ejemplo gráfico sobre la división de los datos entre train, test y validation

Teniendo esto en cuenta se ha dividido el conjunto de entrenamiento y test con una división del 20% y posteriormente se ha recortado un 10% del conjunto de entrenamiento para formar el de validación.

Tras aclarar estos conceptos, los resultados obtenidos en los modelos de aprendizaje automático son los siguientes:

	Apariciones	**Frecuencias**
M. Naive Bayes	0.765	0.74
L. Regression	0.8075	0.7992
L. Support Vector Machine	0.8012	**0.8267**
Random Forest	0.7725	0.76
Decision Tree	0.5812	0.57
Bagging Classifier	0.6947	0.67
Adaboost	0.2337	0.242
Gradient Boosting	0.7022	0.7077
Stochastic Gradient Descent	0.7957	0.762

Tabla 2: Resultados de los modelos de aprendizaje automático para la clasificación

Como pequeña reflexión a aportar se podría decir que, aunque la guía recomendase trabajar con frecuencias en lugar de apariciones, las mejoras son mínimas e incluso hay casos en los que los resultados son peores.

Es destacable que todos los entrenamientos están hechos con exactamente la misma instancia y mismo shuffle[17]del dataset, por lo que las posibles mejoras del cambio a frecuencias son reales y no se deben a un segundo shuffle del dataset.

En cuanto a los resultados, el modelo que mejor resultado proporciona es Linear Support Vector Machine cuando trabaja con frecuencias, se puede encontrar una pequeña explicación a cerca de cómo trabaja el modelo en el apartado 2.2.3.2.1.

Los resultados obtenidos en los modelos de aprendizaje profundo son los siguientes:

[17] Agitaciones del dataset que pueden provocar pequeñas variaciones en la validación del modelo.

	Acierto
ANN 1	**0.837**
ANN 2	0.79
ANN con Word Embedding 1	0.79
ANN con Word Embedding 2	0.77
ANN con Word Embedding 3	0.77
ANN con Embedding de terceros 1	0.19
ANN con Embedding de terceros 1 modificable	0.72
ANN con Embedding de terceros 2	0.42
ANN con Embedding de terceros 2 modificable	0.80
LSTM 1	0.66
LSTM 2	0.75
LSTM 3	0.72
CNN 1	0.79
CNN 2	0.75
CNN con Embedding de terceros 2	0.81
CNN con Embedding de terceros 2 modificable	0.82

Tabla 3: Resultados de los modelos de aprendizaje profundo para la clasificación

La reflexión por realizar en esta sección es bastante profunda y significativa.

Por un lado, se ha encontrado un modelo mejor que el de Linear Support Vector Machines con un porcentaje de acierto del 83.7%. Partiendo de ese modelo y el escalado de redes neuronales con el fin de optimizar los resultados [53] [54], se ha escalado hacía arriba ese mismo modelo sin conseguir mejorar el resultado.

Después, en cuanto a las redes neuronales con Word Embeddings propias, no se ha conseguido mejorar el resultado en ninguna de las variantes probadas (Flatten después del Embedidng, MaxPool y una combinación), siendo el mejor de un 79% de acierto.

Tras conseguir un Embedding de terceros coincidente en un 47% con el vocabulario del dataset (cuanto más alto sea este porcentaje, mejor) se consiguió un triste resultado de 19% de acierto. Aunque es destacable la mejora del modelo a un 72% cuando se modifica la red neuronal y se configura para que modifique ese Embedding en el proceso de entrenamiento según le convenga.

Seguidamente, se consiguió un Embedding mejor coincidente en un 75% con el vocabulario del dataset y el resultado mejoró hasta un 42% de acierto, 80% si se le indica que modifique el embedding durante el entrenamiento, no mejorando el resultado obtenido por la ANN principal.

Teniendo en cuenta lo explicado sobre las LSTMs en el apartado 2.2.3.3.4, el funcionamiento de estas era esperanzador. El resultado en cambio ha sido el peor de todos los tipos de neuronas probadas, proporcionando un 0.75 en el mejor de los casos y de un 0.66 y 0.72 en el de sus variantes con diferentes métodos de regulación.

Siguiendo con las CNNs, estas, aunque no han demostrado el mejor resultado, tampoco ha sido el peor. Esto ha provocado realizar un pequeño experimento juntando los dos tipos de

neuronas que mejor resultado han proporcionado con el objetivo de comprobar si el modelo de ANN puede ser superado.

Es por ello por lo que se ha montado una CNN con un Word Embedding de terceros mejor (75% de coincidencia con el vocabulario del dataset) dando como resultado un 81% de acierto. Teniendo en cuenta la mejora del modelo anterior cuando se modificó para que ajustase el Embedding durante el entrenamiento, se aplicó la misma configuración, aunque esta vez solo se ha conseguido aumentar su acierto de un 81% a un 82%, siendo así nulo el experimento.

Resumiendo, tras probar 3 tipos de neuronas y combinaciones se ha logrado encontrar el mejor modelo, siendo este una ANN con 83.7% de acierto.

En cuanto a los modelos de aprendizaje no supervisado, se ha implementado K-Means y estos son algunos de los clusters:

	Palabras significativas dentro del cluster
Cluster 1:	puerto portuario contenedor autoridad trafico portuaria terminal ferroviario tonelada muelle
Cluster 2:	decoracion color espacio mueble cocina casa amazon hogar navidad pared
Cluster 3:	paciente medico salud enfermedad hospital virus persona caso pandemia riesgo
Cluster 4:	caso comunidad contagio dia sanidad incidencia semana medida salud persona
Cluster 5:	marca consumidor empresa cliente mercado producto digital usuario marketing social
Cluster 6:	ano nuevo pais empresa sector proyecto persona agua energia medida
Cluster 7:	ventana clic haz pinterest telegram whatsapp share correo click amigo
Cluster 8:	partido equipo jugador gol liga club real primero madrid punto
Cluster 9:	vacuna dosis vacunacion pfizer primero persona salud poblacion pais moderna
Cluster 10:	receta aceite plato horno ingrediente salsa minuto sal queso patata
Cluster 11:	coche electrico motor vehiculo cv motorpasion bateria modelo hibrido audi
Cluster 12:	nuevo ano disco camara cancion dia fotografia video primero mejor
Cluster 13:	eta trump derecho ano estado mujer gobierno tribunal politico ley
Cluster 14:	euro millon ano empresa sector cookie espana cookies mes dolar

Cluster 15:	ciudad viaje playa isla lugar siglo hotel pueblo castillo ruta
Cluster 16:	transporte vehiculo camion mercancia carretera conductor transportista reino servicio flota
Cluster 17:	obra arte artista museo exposicion pintura libro premio artistico trabajo
Cluster 18:	ao ms espaa est ltimo econmico club prximo jugador ftbol
Cluster 19:	pelicula serie cine personaje netflix actor you historia marvel temporada
Cluster 20:	gobierno pp sanchez bildu psoe partido presupuestos erc presidente pedro

Tabla 4: Resultado de K-Means para la clasificación

Como se puede observar, algunos de los clusters sí que tienen coherencia (el cluster 8 está claramente relacionado con el deporte, el 10 con la gastronomía y el 20 con la política). Y el índice de Silhouette da como resultado 0,0, lo que es significativo ya que indica que los clusters no están nada diferenciados. Esto es bastante habitual ya que al final al tratar la clasificación de textos, todos comparten una gran parte de vocabulario, lo que perjudica a la hora de agrupar los artículos.

Tras consultar con el experto en la universidad, se llegó a la conclusión de que, aunque se use clustering para clasificar y así tener dos soluciones al mismo problema de clasificación, la comparativa de un modelo supervisado frente a uno no supervisado no es justa ni equitativa, ya que el supervisado tiene una información valiosísima que no tiene el no supervisado.

Por ello, se ha decidido no invertir tiempo en la implementación de DBSCAN y mejorar la clasificación supervisada, dando como resultado la reflexión de líneas más arriba.

Aunque es destacable que este tipo podría resultar útil en un futuro para la recomendación de noticias similares a la leída por el usuario, en lugar de para la clasificación en si misma.

Para finalizar, se profundizará sobre el modelo que mejor resultado ha proporcionado, realizado con ANN.

2.2.4.1 Análisis del modelo que mejor resultado proporciona

El modelo que mejor resultado proporciona es una simple red realizada mediante Keras que contiene las siguientes capas:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 64)                3200064
_________________________________________________________________
activation_7 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 20)                1300
_________________________________________________________________
activation_8 (Activation)    (None, 20)                0
=================================================================
Total params: 3,201,364
Trainable params: 3,201,364
Non-trainable params: 0
```

Ilustración 27: Resumen de la ANN formada para la clasificación

Y su matriz de confusión es la siguiente:

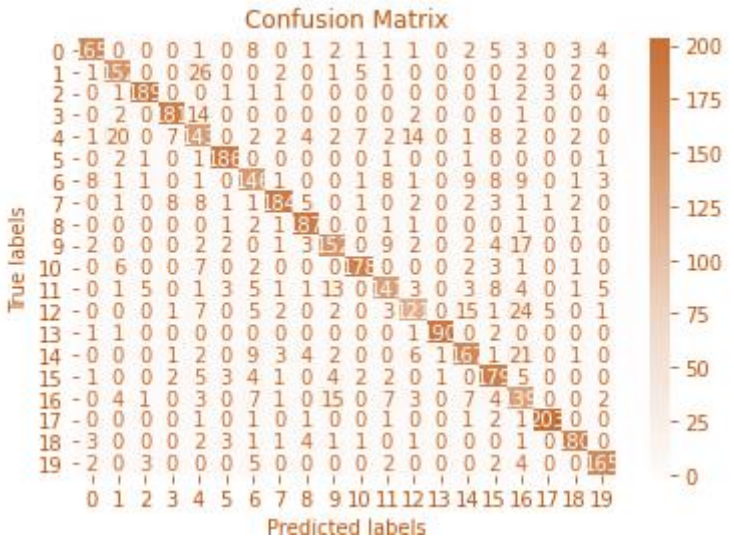


Ilustración 28: Matriz de confusión del modelo de clasificación con mejor resultado

Siendo los temas representados con estos valores numéricos:

- 0: Comunicación y publicidad
- 1: Arte
- 2: Motor
- 3: Cine
- 4: Cultura
- 5: Decoración
- 6: Economía
- 7: Moda
- 8: Alimentación, cocina y gastronomía
- 9: Medicina
- 10: Música
- 11: Naturaleza, medio ambiente y ecología
- 12: Opiniones
- 13: Fotografía
- 14: Política y sindicatos
- 15: Ciencia y tecnología
- 16: Sociedad
- 17: Deportes
- 18: Turismo y viajes
- 19: Transporte

Teniendo esto en cuenta, se puede observar como el modelo principalmente confunde categorías muy relacionadas como pueden ser el arte y la cultura. Además, la situación pandémica que vive la sociedad actualmente hace que la categoría medicina y sociedad estén altamente relacionadas. Aunque estos errores no deberían considerarse errores o fugas como tal ya que incluso un humano tendría dudas al clasificar una noticia de arte como cultura.

Por otro lado, el hecho de incluir en el dataset categorías como opinión o sociedad hace que la eficiencia del modelo se resienta, ya que los contenidos correspondientes pueden tratar otros temas.

Asimismo, se puede observar como el modelo diferencia correctamente aquellas temáticas con un vocabulario muy concreto y sin otras categorías influyentes. Estas son el deporte, la fotografía, moda o alimentación, entre otras.

Para finalizar, a modo de curiosidad, el hecho de que existan un mayor número de artículos correspondientes a cada categoría en el conjunto de datos hace que el modelo mejore. Aunque se es consciente de que se llegará a un punto en el que la mejora sea nula, en el momento de escribir estas líneas el dataset y contiene mil artículos únicos por categoría y aunque se sospecha la proximidad a ese límite, aún no se ha alcanzado.

A continuación, se puede como a medida que el modelo contiene más artículos por categoría, es más decidido hacía una categoría concreta. Esto es debido a que tiene más pruebas o muestras de cada una de las categorías, lo que le permite seleccionar nuevas palabras o dar más peso a las antiguas para lograr el objetivo.

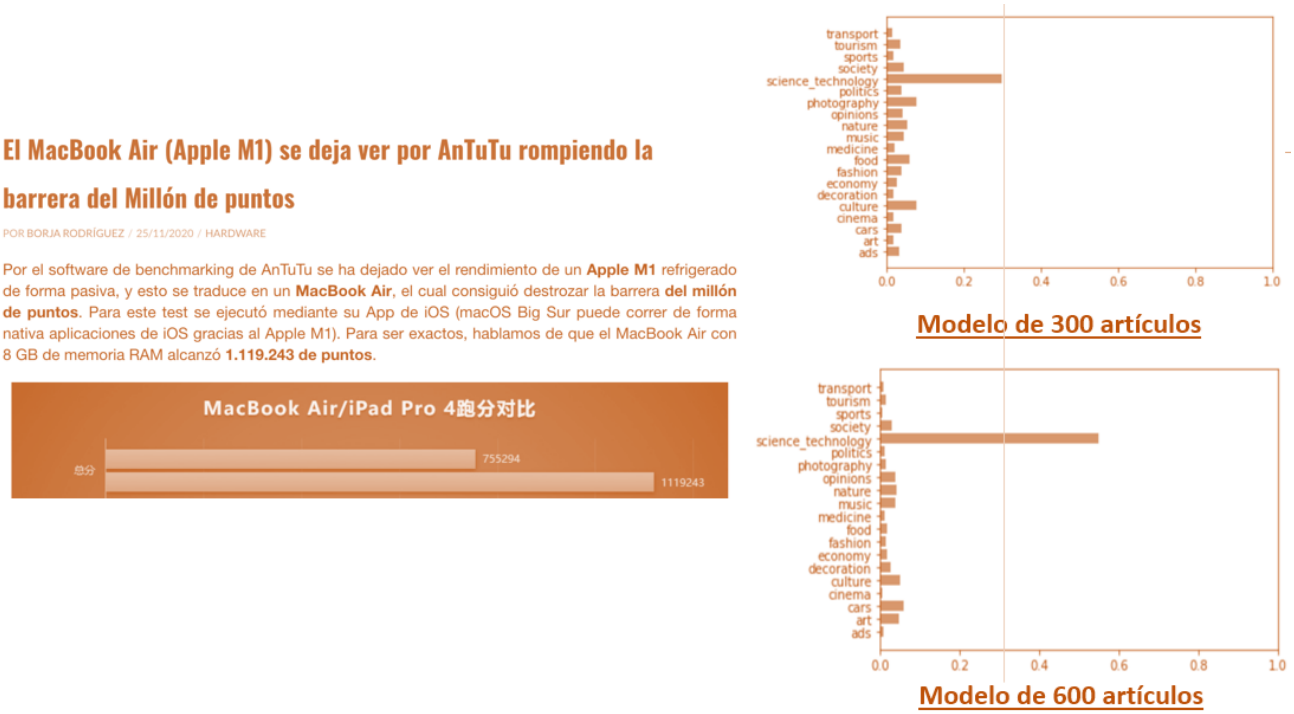


Ilustración 29: Ejemplo 1 sobre la mejoría del modelo

Ilustración 30: Ejemplo 2 sobre la mejoría del modelo

En el siguiente apartado se investigará sobre del clasificador.

2.3 Desarrollo del filtrador

A lo largo de este apartado se detallará la investigación realizada sobre cuál es el modelo que mejor se adapta a los datos, con el fin de obtener un filtrador automático que funcione correctamente.

2.3.1 Introducción

Una de las claves para que Tentu, la revista electrónica personalizada, funcione y los usuarios lo utilicen de forma habitual es que el contenido mostrado en ella sea reciente y actualizado cada vez que el usuario acceda a la revista. Así, podrá ver las noticias que han sido publicadas hace escasos minutos y saber en cualquier momento lo que ocurre con relación a las temáticas y los pueblos favoritos del usuario.

Como se ha mencionado anteriormente, la recolección de los contenidos que se muestran en Tentu es realizada mediante dos formas. Una es automática y se responsabiliza de la colecta de la mayoría de los datos disponibles para los usuarios mediante fuentes RSS, y otra manual disponible para los usuarios con roles de creador de contenido.

En cuanto a los contenidos recolectados de manera automática, estos proveen la mayoría del contenido disponible, ya que se recolecta contenido de más de 700 fuentes RSS en euskera

y castellano. Algunas de ellas provienen de ayuntamientos de pueblos o fuentes de información fiables como periódicos, pero también se incluyen fuentes RSS de asociaciones menos conocidas y blogs de personas con una menor fiabilidad.

Esto puede ser un problema grave para el usuario final, ya que puede encontrarse con artículos o eventos recolectados automáticamente que contengan elementos atentatorios a la imagen y privacidad de las personas, incorporen actividades ilegales o, simplemente, que no resulten cómodas y amenas para la vista, por ejemplo, haciendo un uso excesivo de las mayúsculas.

Además del contenido recolectado automáticamente, los creadores de contenido también pueden crear artículos y eventos, y estos también pueden contener elementos inadecuados que no deberían de ser mostrados a los usuarios finales, ya que pueden resultar ofensivos y provocarían que la experiencia de uso de la aplicación no fuese apropiada.

Para solventar este problema, se ha detectado la necesidad de un sistema de detección y filtrado de los contenidos que son introducidos en el sistema.

2.3.2 Problema a afrontar

Teniendo en cuenta la introducción sobre el mundo de inteligencia artificial realizado en el apartado 2.2.1, el objetivo de esta sección es el de identificar el tipo de problema a afrontar para el desarrollo de un modelo que sea capaz de censurar textos.

Como se ha indicado en la contextualización inicial (apartado 1.1), GAIA es un conjunto de servicios tanto de clasificación como de filtrado de contenidos en la nube. El principal objetivo de este proyecto es el de facilitar y agilizar la selección de palabras tanto ofensivas como no mediante un modelo de inteligencia artificial.

En cuanto al filtrado o censura de contenidos, sería adecuado que el modelo proporcionase al cliente un grado de censura para que sea la persona usuaria la que pueda decidir qué hacer con el contenido enviado al servicio en base al resultado.

Es por ello por lo que se va a afrontar este problema como uno de clasificación binaria, siendo la probabilidad de que sea censurable (valor entre 0 y 1) el grado de censura citado en el párrafo anterior.

2.3.3 Implementación

En esta sección se detallarán los datos, explicando su origen y tratamiento, así como los algoritmos. Finalmente, se reflexionará sobre los resultados obtenidos.

2.3.3.1 Información y tratamiento de los datos

La primera labor a realizar ha sido la búsqueda de diferentes datasets que facilitasen la identificación de palabras ofensivas, al estar todas en inglés, el equipo de desarrollo decidió formar uno propio a través de la captación de contenidos de diversas fuentes.

Es por ello por lo que, mediante la localización de un conjunto de ficheros, sin un formato definido, pero si con un patrón claro que permitió al equipo de desarrollo extraer un gran número de textos de la Wikipedia con mucho contenido clasificado como "no-ofensivo".

Una vez hecho esto el principal problema era la búsqueda de contenido ofensivo con el objetivo de añadirlo al conjunto de datos final. Esta tarea resultó especialmente complicada ya que gran parte de las aplicaciones web en las que los usuarios tienen la opción de redactar sus opiniones o comentarios cuentan con un sistema de filtrado o moderación propia, por lo que no iban a aparecer opiniones ofensivas. Por este motivo aplicaciones como TripAdvisor o Airbnb quedan totalmente descartadas.

En el momento de afrontar el problema, el equipo de desarrollo contaba con un listado de unas 500 palabras malsonantes del desarrollo de una funcionalidad de GAIA, por lo que se optó por hacer uso de ella para desarrollar un script que tomase cada una de esas palabras y consultase a través de la API [18]de Twitter diferentes Tweets [19]que contuvieran al menos una palabra malsonante.

De este modo se ha obtenido muchísimo contenido filtrable con calidad contrastada, ya que para que los Tweets fueran aptos para ser incluidos en el dataset, estos debían cumplir unos requisitos como la longitud mínima de caracteres.

Tras realizar completar los desarrollos se alcanzó el siguiente dataset:

DATASET	NOT OFFENSIVE	OFFENSIVE	TOTAL
WIKIPEDIA	189.990 (92.33%)	18.378 (7.67%)	205.766 (100%)
TWITTER	1.945 (6.44%)	28.235 (93.55%)	30.180 (100%)
COMBINACIÓN	191.935 (80.25%)	46.613 (19.75%)	235.946 (100%)

Tabla 5: Formación del dataset de filtrado

Cabe destacar que se han considerados como ofensivos, aquellos artículos de Wikipedia que contuvieran cinco o más palabras malsonantes del listado anteriormente citado.

Tras una jornada de reflexión, se observó que el contenido proveniente de Wikipedia era mucho mayor a la de Twitter y que alcanzarlo iba a suponer mucho tiempo. Además, el temor de que el modelo internamente hiciera una especie de clasificación dependiendo de la longitud del texto, hizo que el equipo de desarrollo decidiera crear un equilibrador similar al del modelo de clasificación temática de modo que se tengan la misma cantidad de contenidos de cada uno de los orígenes y clases.

Después de formar el dataset, se debe preparar para el entrenamiento del modelo. En problemas de procesamiento del lenguaje natural, como este o el de clasificación temática de la

[18] Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.
[19] Un Tweet es el nombre que reciben las publicaciones en la red social Twitter.

sección anterior, es necesario realizar una pequeña limpieza del texto con el fin de que el aprendizaje sea más sencillo, tal y como se ha visto en el apartado 2.2.3.1. Resumiendo, estas son algunas de las acciones que se han realizado:

- Eliminar las palabras sin importancia para el filtrado de textos como pueden ser en el caso de los Tweets las citas a otros usuarios o referencias a diferentes Hashtags.
- Eliminar los signos de puntuación
- Eliminar las acentuaciones y caracteres propios como la ñ
- Lematizar para obtener el núcleo de cada palabra

Una vez adaptado el dataset, el tratamiento dado a los datos es exactamente el mismo que en el modelo del clasificador temático, es decir se ha tokenizado el dataset. Para más información sobre el proceso de tokenización se recomienda la lectura del apartado 2.2.3.1 de este mismo documento.

2.3.3.2 Información sobre los algoritmos

En cuanto a los algoritmos utilizados para el entrenamiento de los modelos, son exactamente los mismos que en la clasificación temática de textos. La única variación que tienen es que la última capa de las redes neuronales de los algoritmos de aprendizaje profundo está ajustada para que la salida sea la adecuada para una clasificación binaria, como el problema que ocupa a esta sección.

2.3.3.3 Entorno utilizado en el entrenamiento

Tal y como se ha explicado en el apartado 2.2.3.5 de este documento, los equipos de desarrollo no cuentan con la potencia suficiente por lo que al igual que en la clasificación de textos, se hará uso de Google Colab [51].

2.3.4 Resultados obtenidos

Para la validación del modelo se va a seguir el mismo procedimiento aplicado en la clasificación de textos explicado en el apartado 2.2.4.

Tras aclarar estos conceptos, los resultados obtenidos en los modelos de aprendizaje automático son los siguientes:

	Apariciones	Frecuencias
M. Naive Bayes	0.69	0.70
L. Regression	0.82	0.83
L. Support Vector Machine	0.80	0.85
Random Forest	0.86	**0.86**
Decision Tree	0.77	0.79
Bagging Classifier	0.81	0.82
Adaboost	0.64	0.65
Gradient Boosting	0.63	0.66
Stochastic Gradient Descent	0.76	0.69

Tabla 6: Resultados de los modelos de aprendizaje automático para el filtrado

Como pequeña reflexión a aportar se podría decir que, al igual que en la clasificación temática, aunque la guía recomendase trabajar con frecuencias en lugar de apariciones, las mejoras son mínimas e incluso hay casos en los que los resultados son peores.

Es destacable que todos los entrenamientos están hechos con exactamente la misma instancia y mismo shuffle[20] del dataset, por lo que las posibles mejoras del cambio a frecuencias son reales y no se deben a un segundo shuffle del dataset.

En cuanto a los resultados, el modelo que mejor resultado proporciona es Random Forest, con un 86% de acierto, siendo indiferente el hecho de trabajar con apariciones o frecuencias.

Los resultados obtenidos en los modelos de aprendizaje profundo son los siguientes:

	Acierto
ANN 1	**0.8842**
ANN 2	0.8367
ANN con Word Embedding 1	0.79
ANN con Word Embedding 2	0.85
ANN con Word Embedding 3	0.84
ANN con Embedding de terceros 1	0.73
ANN con Embedding de terceros 1 modificable	0.85
ANN con Embedding de terceros 2	0.76
ANN con Embedding de terceros 2 modificable	0.86
LSTM 1	0.50
LSTM 2	0.60
LSTM 3	0.65
CNN 1	0.85
CNN 2	0.81
CNN con Embedding de terceros 2	0.84
CNN con Embedding de terceros 2 modificable	0.85

Tabla 7: Resultados de los modelos de aprendizaje profundo para el filtrado

La reflexión a realizar es similar a la de la clasificación temática.

[20] Agitaciones del dataset que pueden provocar pequeñas variaciones en la validación del modelo.

Se ha encontrado un modelo mejor que el de Random Forest con un porcentaje de acierto del 88.42%. Partiendo de ese modelo y siguiendo el escalado anteriormente citado, no se ha conseguido mejorar el resultado.

Por otro lado, en cuanto a las redes neuronales con Word Embeddings propias, al igual que en la clasificación textual, no se ha conseguido mejorar el resultado en ninguna de las variantes probadas (Flatten después del Embedidng, MaxPool y una combinación), siendo el mejor de un 85% de acierto.

Tras conseguir un Embedding de terceros coincidente en un 21% con el vocabulario del dataset (cuanto más alto sea este porcentaje, mejor) se consiguió un resultado de 73% de acierto. Aunque el modelo ha mejorado a un 85% cuando se modifica la red neuronal y se configura para que modifique ese Embedding en el proceso de entrenamiento según le convenga.

Seguidamente, se consiguió un Embedding mejor coincidente en un 41% con el vocabulario del dataset y el resultado mejoró hasta un 76% de acierto, 86% si se le indica que modifique el embedding durante el entrenamiento, no mejorando el resultado obtenido por la ANN principal.

Al igual que en el modelo de clasificación temática, el resultado de los modelos de LSTM ha sido el peor de todos los tipos de neuronas probadas, proporcionando un 0.65 en el mejor de los casos y de un 0.50 y 0.60 en el de sus variantes con diferentes métodos de regulación.

En cuanto a las CNNs, aunque no han demostrado el mejor resultado, tampoco ha sido el peor. Esto ha provocado realizar exactamente el mismo experimento que en el modelo anterior.

Al igual que en la clasificación temática, se ha montado una CNN con un Word Embedding de terceros mejor (41% de coincidencia con el vocabulario del dataset) dando como resultado un 84% de acierto. Teniendo en cuenta la mejora del modelo anterior cuando se modificó para que ajustase el Embedding durante el entrenamiento, se aplicó la misma configuración, aunque esta vez solo se ha conseguido aumentar su acierto de un 84% a un 85%, siendo así nulo el experimento.

Resumiendo, tras probar 3 tipos de neuronas y combinaciones se ha logrado encontrar el mejor modelo, siendo este una ANN con 88.42% de acierto.

En cuanto a los modelos de aprendizaje no supervisado, se ha implementado K-Means sin conseguir ningún resultado mostrable ya que son totalmente incoherentes. Además, teniendo en cuenta lo indicado en el apartado anterior respecto a la comparación directa entre un algoritmo de aprendizaje supervisado se ha decidido no invertir tiempo ni realizar esfuerzos en ajustar el algoritmo hasta conseguir algo medianamente coherente. Por lo que en la siguiente sección se analizará el algoritmo que mejor resultado proporciona.

2.3.4.1 Análisis del modelo que mejor resultado proporciona

El modelo que mejor resultado proporciona, al igual que en la clasificación de textos, es una simple red realizada mediante Keras que contiene las siguientes capas:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                multiple                  6400128
_________________________________________________________________
activation (Activation)      multiple                  0
_________________________________________________________________
dropout (Dropout)            multiple                  0
_________________________________________________________________
dense_1 (Dense)              multiple                  129
_________________________________________________________________
activation_1 (Activation)    multiple                  0
=================================================================
Total params: 6,400,257
Trainable params: 6,400,257
Non-trainable params: 0
_________________________________________________________________
```

Ilustración 31: Resumen de la ANN formada para el filtrado

Y su matriz de confusión es la siguiente:

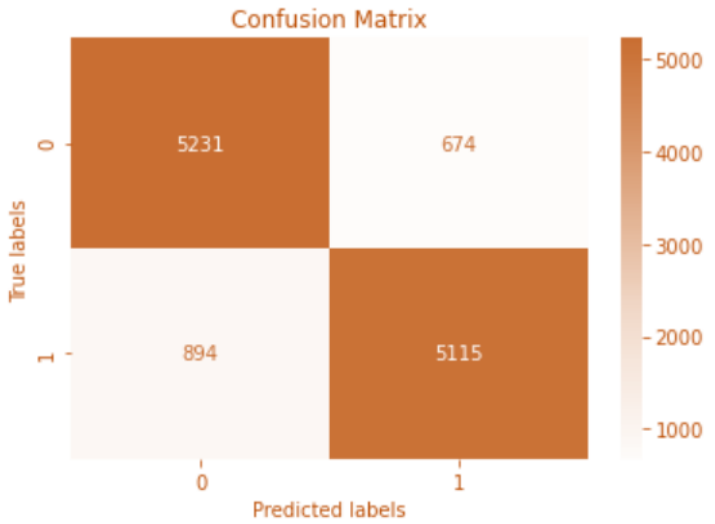


Ilustración 32: Matriz de confusión del modelo de filtrado con mejor resultado

Observando la matriz se puede comprobar que efectivamente el modelo funciona bien, por lo que es hora de ponerlo en contexto.

Si se le introducen textos con modificaciones malsonantes como las siguientes:

Jenson Button, campeón del mundo de Fórmula 1 en 2009, ha analizado desde un punto de vista muy peculiar la llegada de Carlos Sainz a Ferrari. "Desde luego que **si el hijo de puta** de Binotto ha ido a por Carlos porque cree que tiene un buen buenas manos y era una buena oportunidad, yo creo que se ha equivocado, porque creo que Carlos es un ganador, es como su **puto padre y su puta madre y me cago en dios y la concha de la lola",** afirma en una videocharla con Sky Sports en referencia al bicampeón del mundo de rallies y tricampeón del Dakar y a la elección de Sainz en vez de Ricciardo.	→ 0.96
Hace 13 años, Irrational Games nos pilló a todos desprevenidos con BioShock, un **puto juegazo** de acción en primera persona que se convirtió desde el mismo día de su lanzamiento en toda una obra maestra de este hobby que tanto nos apasiona. Su **puta mierda de ambientación** en una decadente ciudad sumergida bajo el mar, su gusto por el detalle, su apabullante y opresivo diseño artístico, su forma de usar el lenguaje propio de los videojuegos para narrarnos una historia fantástica de ciencia ficción que toca temas muy actuales, su manera de poner a prueba nuestra moralidad y sus múltiples posibilidades a la hora de combatir usando armas, poderes y el mismísimo entorno son solo algunas de sus muchísimas virtudes. Una maravilla atemporal que sigue poniéndonos los pelos de punta a día de hoy y que dio origen a dos secuelas, una desarrollada por 2K Marin y otra por la mismísima Irrational Games, siendo esta última otra **basura para los huevudos de los fan**s de la acción y los videojuegos en general.	→ 0.80
Hola que tal me llamo Ander y **me cago en la concha de tu madre**	→ 0.96

Se puede observar cómo produce índices de censura altos, lo cual está bien, probando con textos sin ningún tipo de modificación:

Jenson Button, campeón del mundo de Fórmula 1 en 2009, ha analizado desde un punto de vista muy peculiar la llegada de Carlos Sainz a Ferrari. "Desde luego que si Binotto ha ido a por Carlos porque cree que tiene un buen buenas manos y era una buena oportunidad, yo creo que se ha equivocado, porque creo que Carlos es un ganador, es como su padre", afirma en una videocharla con Sky Sports en referencia al bicampeón del mundo de rallies y tricampeón del Dakar y a la elección de Sainz en vez de Ricciardo.

→ 0.006

Hace 13 años, Irrational Games nos pilló a todos desprevenidos con BioShock,un juegazo de acción en primera persona que se convirtió desde el mismo día de su lanzamiento en toda una obra maestra de este hobby que tanto nos apasiona.

Su logradísima ambientación en una decadente ciudad sumergida bajo el mar, su gusto por el detalle, su apabullante y opresivo diseño artístico, su forma de usar el lenguaje propio de los videojuegos para narrarnos una historia fantástica de ciencia ficción que toca temas muy actuales, su manera de poner a prueba nuestra moralidad y sus múltiples posibilidades a la hora de combatir usando armas, poderes y el mismísimo entorno son solo algunas de sus muchísimas virtudes. Una maravilla atemporal que sigue poniéndonos los pelos de punta a día de hoy y que dio origen a dos secuelas, una desarrollada por 2K Marin y otra por la mismísima Irrational Games, siendo esta última otra aventura imprescindible para los fans de la acción y los videojuegos en general.

→ 0.007

Hola que tal soy Ander, espero que estés muy bien

→ 0.09

Como se puede ver, los índices se corresponden al contenido. La única reflexión que se puede hacer en este caso es que al parecer los índices que proporciona el modelo son demasiado extremistas y puede tener relación con que los datos del dataset están clasificados en binario, es decir, 0 o 1.

Quizás clasificar diferentes frases o palabras en una escala entre 0, 0.25, 0.5, 0.75 y 1 pueda ayudar a que el modelo proporcione resultados menos extremistas. En todo caso, a nivel informático, es decir, en cuanto a la programación del modelo, no se tendría que cambiar nada ya que lo único a modificar son los datos.

Tras seleccionar el mejor modelo tanto para el filtrado como para la clasificación de contenidos, es hora de detallar el desarrollo y despliegue del software que los proporcione a los clientes.

2.4 Desarrollo y despliegue de software que ofrezca los modelos

Tras desarrollar y seleccionar el modelo que mejor resultados proporcionaba es el momento de elegir el método mediante el cual proporcionar a los clientes la posibilidad de hacer consultas al modelo.

Resulta interesante contemplar que el desarrollo sea útil para ambos modelos, por lo que se debe generar de la forma más genérica o abstracta posible.

Después de consultar otras opciones en el mercado y reflexionar sobre desarrollos pasados de algunos de los miembros del equipo, se llegó a la conclusión de que la mejor opción pasaba por realizar un servicio REST. Además, el equipo contaba con experiencia en este tipo de servicios tal y como se ha visto en el desarrollo del recopilador, por lo que la barrera del desconocimiento de la tecnología era inexistente.

Es por ello por lo que se ha desarrollado una aplicación Flask en Python. Su principal cometido es compilar el modelo guardado en un fichero H5, recoger tanto el tokenizador como el encoder y abrir un path mediante el cual la persona usuaria consulta al servicio adjuntando un texto a filtrar o clasificar.

Debido a la experiencia ganada mediante el desarrollo del recopilador, se han Dockerizado los softwares de consulta, de modo que sean totalmente agnósticos a la plataforma de despliegue, dando libertad para seleccionarla.

Al ser este un software que estará en constante cambio debido a posibles mejoras del modelo bien sea por un mejor ajuste de parámetros o adición de más textos al conjunto de datos, es interesante aplicar integración y despliegue continúo.

2.4.1 Integración y despliegue continúo

En el ámbito del desarrollo y despliegue del software cada vez es más común emplear técnicas que favorezcan la comunicación e integración de los desarrolladores. Por tanto, es vital aplicar conceptos como la integración, entrega o despliegue continúo [55].

Hoy en día, las empresas se centran en proporcionar a sus clientes un software con la máxima calidad. Para ello, emplean métodos y técnicas que garanticen dicha calidad y eviten riesgos o pérdidas económicas.

En este sentido, entran en juego nuevos paradigmas de organización que faciliten y agilicen el flujo de trabajo y promuevan la automatización de los procesos.

Para solventar estas necesidades surge la integración continúa, que constituye uno de los pilares fundamentales sobre los que se sustenta el desarrollo de software.

La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas [56]. Su principal objetivo es el de encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software.

Con la integración continua, los desarrolladores envían los cambios de forma periódica a un repositorio compartido con un sistema de control de versiones como Git. Antes de cada envío, estos pueden elegir ejecutar pruebas de unidad local en el código como medida de verificación adicional antes de la integración. Un servicio de integración continua crea y ejecuta automáticamente pruebas de unidad en los nuevos cambios realizados en el código para identificar inmediatamente cualquier error.

La entrega continua es una práctica de desarrollo de software posterior a la integración continúa mediante la cual se preparan automáticamente los cambios en el código y se entregan a la fase de producción [57].

La entrega continua amplía la integración continua al implementar todos los cambios en el código en un entorno de pruebas o de producción después de la fase de compilación. Cuando la entrega continua se implementa de manera adecuada, los desarrolladores dispondrán siempre de un artefacto listo para su implementación que se ha sometido a un proceso de pruebas estandarizado.

La diferencia entre la entrega y la implementación continua es la diferencia de aprobación manual para actualizar la producción. Con la implementación continua, la producción tiene lugar de manera automática, sin aprobación explícita.

La entrega continua automatiza todo el proceso de publicación de software. Cada revisión efectuada activa un proceso automatizado que crea, prueba y almacena la actualización. La decisión definitiva de implementarla en un entorno de producción en vivo la toma el desarrollador.

En la siguiente ilustración se puede observar un resumen de lo explicado:

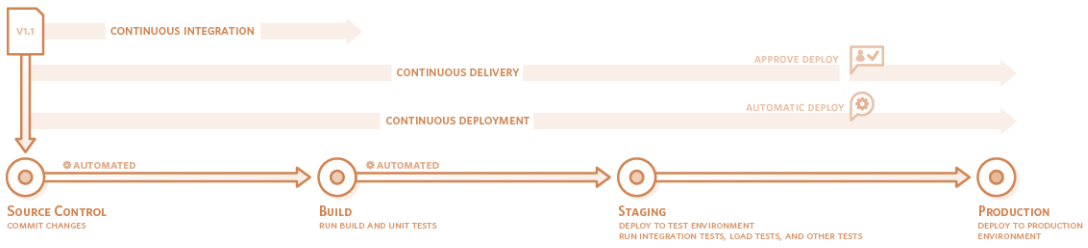


Ilustración 33: Resumen sobre el proceso seguido cada vez que ocurre una actualización de código en el repositorio

2.4.1.1 Ventajas que suponen adoptar estas prácticas

Las ventajas que suponen adoptar este tipo de prácticas son las siguientes

- Propicia la entrega inmediata del código. Es decir, no es necesario acumular funcionalidades o desarrollo autónomos.
- Ahorra costes y esfuerzos (se automatizan)
- Aumenta la confianza en el momento de desplegar el código hacia la etapa de producción.
- Simplifica el trabajo en equipo.
- Posibilita la identificación temprana de los errores.

En definitiva, la aplicación de estas técnicas se traduce en flexibilidad a la hora de incluir cambios en el desarrollo de software y eficiencia en los equipos. A su vez, posibilita la entrega al cliente de un producto de calidad en el menor tiempo posible.

Tras analizar los beneficios de aplicar estas prácticas es hora de detallar los runner y proveedores seleccionados para aplicarlo.

2.4.2 Runner del pipeline y despliegue

Para lograr lo explicado en la sección anterior, se hace uso de diferentes herramientas. Por un lado, un proveedor que gestione todas las etapas del proceso de integración, entrega y despliegue continúo y, por otro lado, otro que permita el despliegue de las aplicaciones.

En este caso, se ha seleccionado como orquestador la propia herramienta del controlador de versiones, Gitlab CI/CD. Como proveedor para el despliegue de la aplicación se ha seleccionado Heroku, ya que es realmente sencilla su implementación en el pipeline mediante herramientas como DPL[21].

Además, resulta interesante citar, que como la aplicación ha sido contenedorizada, se va a hacer uso del propio registro de contenedores que Gitlab proporciona, con el fin de tener etiquetado y organizado cada una de las versiones del software. De este modo, en caso de que el software contenga algún error no contemplado, sería posible volver a una versión funcional en cuestión de minutos.

El flujo de trabajo establecido para cada actualización del código se ha dividido en 4 etapas y son las siguientes:

- Build: Descarga del registro de contenedores de Gitlab el contenedor con tag latest, con el fin de utilizarlo como caché y ahorrar tiempo. Después, se construye el nuevo contenedor con el nuevo aporte de código y se sube al registro de contenedores de Gitlab con el tag TEST.
- Test: La segunda parte del pipeline se loguea en el registro de contenedores de Gitlab y descarga y ejecuta el contenedor formado en la etapa anterior con tag TEST. Una vez el servicio está en marcha se pasa un test básico de integración y se aplica el formateador de código automático BLACK.

[21] Dpl es una herramienta de despliegue hecha para el despliegue continuo que es desarrollada y utilizada por Travis CI, pero también puede ser utilizada con GitLab CI/CD.

- Delivery: La tercera parte del pipeline se encarga de descargar las imágenes de TEST una vez más (con la diferencia de que ahora se tiene la certeza de que la aplicación funciona con el nuevo aporte de código) etiquetándolas como RELEASE y LATEST y las sube al registro. De este modo son tres los contenedores que quedan por cada commit que pase el pipeline: TEST, RELEASE y LATEST.
- Deploy: Recoge el contenedor con etiqueta RELEASE y lo redespliega en el proveedor Heroku, terminando ahí la ejecución del pipeline.

En el momento de escribir estas líneas, el equipo de desarrollo es totalmente consciente de que lo ideal sería tener dos despliegues (develop y master, por ejemplo) con el fin de asegurar que la aplicación master está siempre estable. Los motivos por los que no se ha hecho así se ha detallado en el apartado de líneas futuras, en la sección 5.5.

2.5 Desarrollos adicionales

Como se ha explicado en la contextualización, el hecho de que se lleve varios años realizando las prácticas en la misma empresa, provoca que se colabore en otros tipos de desarrollos. Estos pueden ser la propia revista Tentu, en este caso en la adaptación y publicación de la aplicación Flutter al sistema operativo iOS o el producto Evidence Box, en el que he tenido que trabajar en el cambio de proveedor cloud para el back-end.

2.5.1 Adaptación y publicación de la aplicación Flutter a iOS

Uno de los desarrollos principales de ISEA es la revista electrónica personalizada Tentu. El año pasado se comenzó el desarrollo de una aplicación para smartphones con el fin de expandir el público. Como el objetivo era que fuese valido tanto para iOS, el sistema operativo de móviles Apple, como para Android, se seleccionó Flutter como framework de desarrollo [58].

Flutter es un framework de código abierto desarrollado por Google para crear aplicaciones nativas de forma fácil, rápida y sencilla. Su principal ventaja radica en que genera código 100% nativo para cada plataforma, con lo que el rendimiento y la UX es totalmente idéntico a las aplicaciones nativas tradicionales.

En el mercado de desarrollo de aplicaciones móviles para iOS y Android hay una gran cantidad de frameworks o herramientas que permiten utilizar un mismo código fuente para ambas plataformas. Pero ninguna genera código 100% nativo.

La principal y más importante ventaja de Flutter es que se desarrolla un solo proyecto para todos los sistemas operativos, lo que significa una reducción de costes y tiempo de producción.

Estas son algunas de las funcionalidades de Flutter:

- Calidad nativa: Las aplicaciones nativas se desarrollan específicamente para un sistema operativo, Flutter utiliza todas las ventajas de las aplicaciones nativas para conseguir calidad en el resultado final.

- Experiencia de usuario: Flutter incluye Material Design de Google y Cupertino de Apple, con lo que la experiencia de usuario es óptima y los interfaces de usuario idénticos a los de las aplicaciones desarrolladas por las propias compañías.
- Tiempo de carga: Una de las principales causas de abandono de una aplicación es el tiempo que tarda en cargar, con Flutter se experimentan tiempos de carga por debajo de un segundo en cualquiera de los soportes iOS o Android.
- Desarrollo ágil y rápido: Gracias a la característica hot-reload, es posible programar y ver los cambios en tiempo real en los simuladores.
- Otra de las ventajas de utilizar este framework es que da igual el sistema operativo que utilices, ya que es posible descargarlo para Windows, Mac o Linux.

Como principal desventaja de utiliza Flutter como framework de desarrollo para aplicaciones de smartphone, está el hecho de tener que utilizar Dart como lenguaje de programación.

Debido al desconocimiento del lenguaje, ha sido necesario un tiempo para conocerlo con un nivel mínimo como para realizar la adaptación.

La adaptación ha consistido principalmente en la modificación de componentes propios de cada sistema operativo como puede ser diferentes selectores de género o numéricos, tal y como se puede observar en la siguiente ilustración:

Ilustración 34: Comparativa entre componentes de Android e iOS

Tras realizar la adaptación de gran parte de los componentes de la aplicación, se ha procedido a su publicación y actualmente se encuentra disponible para iPhone y iPad en la tienda oficial de Apple.

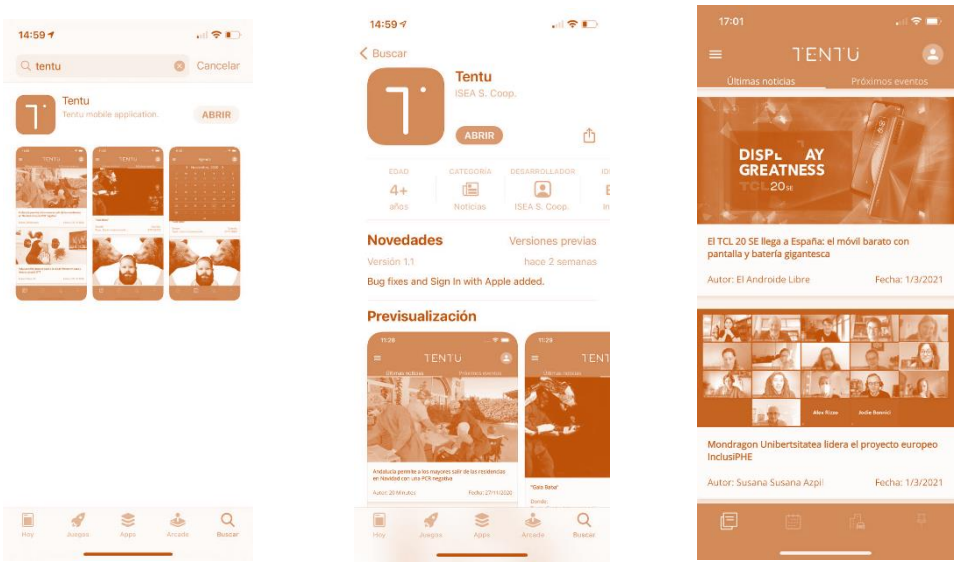


Ilustración 35: Prueba de que la aplicación está publicada en la AppStore de Apple

2.5.2 Cambio de proveedor servidor EvidenceBox

Otro los desarrollos en los que ISEA participa es EvidenceBox. Este, es un servicio por Internet soportado en tecnologías Blockchain que permitirá, vía servicios de confianza en las transacciones electrónicas según el Reglamento Europeo (UE) Nº 910 eIDAS [59], la navegación certificada de sitios web de comercio electrónico o de servicios digitales, posibilitando aportar prueba legal fehaciente de la realización de actos electrónicos que acompañen a los procesos de consulta informativa, utilización de servicios digitales, compra o contratación electrónica de bienes y servicios en la Unión Europea.

ISEA forma parte en el proyecto desarrollando la aplicación web mediante la cual poder informarte acerca del servicio y contratarlo en caso de estar interesado/a. Además, mediante la aplicación web se ofrece la posibilidad de gestionar y visualizar las evidencias generadas, para ello hace uso de un algoritmo generador de video partiendo de imágenes/frames. Este algoritmo es muy costoso computacionalmente y el servidor alojado en AppEngine[22] no era capaz de gestionar todos los procesos de forma fluida, además era excesivamente caro para lo que ofrecía. Es por ello por lo que se decidió migrarlo a otro proveedor con dos objetivos, por un lado, que tenga una mayor capacidad computacional y por otro que sea más económico.

Además, se quiere aprovechar la oportunidad para poder dockerizar el servidor y hacer que sea ajeno al proveedor, evitando así posibles incompatibilidades. El servidor de EvidenceBox está desarrollado en Node.js mediante el framework Express, por lo que haciendo uso de diferentes guías se ha conseguido formar un sistema Docker con dos contenedores:

[22] AppEngine es un servicio de Google que permite crear aplicaciones muy escalables en una plataforma sin servidor totalmente gestionada.

- Uno con nginx con el fin de enlazar el servidor al puerto 433 y aplicar el protocolo SSL[23].
- Otro con el servidor corriendo en Express.js y el gesto de procesos PM2, del cual se hablará más adelante.

En cuanto al proveedor de alojamiento, no se quiso olvidar el contexto académico de este proyecto y se aprovechó la oportunidad para conocer más sobre un proveedor, en este caso AWS Elastic Container Service [60].

Para gestionar mejor las versiones del servidor y aprovechando que el servidor se ha contenedorizado se ha hecho uso del registro de contenedores que la misma Amazon ofrece, AWS Elastic Container Registry.

Tras finalizar la configuración de los diferentes servicios y comprobar que funcionalidades como la generación de video funcionaba correctamente, se ha decidido aplicar Integración y despliegue continúo con el fin de que el servidor se actualice en cada aporte nuevo de código automáticamente.

En este caso, al ser un producto en desarrollo, al igual que los servicios de ofrecimiento del modelo, se ha decidido no duplicar las aplicaciones en entornos de desarrollo y producción y dejarlo para el futuro, dando así la posibilidad de invertir tiempo en otros desarrollos.

2.5.3 Colaboración en el desarrollo y mejora de GAIA

Se debe recordar, que el propósito general de este proyecto es el de añadir valor a GAIA, utilizando inteligencia artificial para seleccionar y ponderar las diferentes palabras en la clasificación y filtrado de textos.

Además, como el proyecto tiene pendiente una supervisión por parte del Ministerio de Economía y Empresa, se han tenido que solucionar diferentes bugs de la aplicación web como puede ser el hecho de que en un caso concreto la aplicación web no enviase la factura a los clientes.

Por otro lado, como el servidor estaba programado con Express.js (al igual que el servidor de EvidenceBox) se ha hecho uso de la experiencia obtenida en la contenedorización y migración para hacer lo mismo con el servidor de GAIA. Aunque esta vez el despliegue se ha hecho en Heroku, ya que por suerte la capacidad de computación requerida es más baja.

En este caso, al ser un producto que ya se encuentra disponible para los clientes (de hecho, Tentu se soporta en GAIA para realizar su clasificación textual) se ha duplicado el servidor en diferentes entornos, por un lado, el de desarrollo y por otro el de producción.

Además, también se ha implementado un pipeline con integración, entrega y despliegue continúa con el fin de agilizar la entrega de la nueva versión al cliente, utilizando Gitlab CI/CD como orquestador. Asimismo, se han realizado testeos de absolutamente todas las

[23] EL SSL proporciona un canal seguro entre dos computadoras o dispositivos que operan a través de Internet o de una red interna.

funcionalidades del servidor para asegurar su correcto funcionamiento en cada uno de los commits. El flujo del commit se ha dividido en 5 etapas y son las siguientes:

- BUILD: Construye el nuevo contenedor con el aporte de código y lo tagea como test.
- TEST: Comprueba si el aporte de código no rompe alguna funcionalidad existente aplicando testeos.
- DELIVERY: Una vez se comprueban los testeos se etiqueta el contenedor de test como release y latest.
- DEPLOY_DEVELOP: En caso de que el aporte de código corra en la rama de desarrollo, solamente se actualiza el servidor de desarrollo.
- DEPLOY_MASTER: En caso de que exista un merge_request para actualizar el servidor de producción con nuevas funcionalidades, se actualiza el servidor master o de producción.

Asimismo, no se ha desperdiciado la oportunidad para añadir PM2 al servidor [61].

PM2 es un gestor de procesos en producción para las aplicaciones Node.js que tiene un balanceador de carga incorporado. PM2 permite mantener siempre activas las aplicaciones y volver a cargarlas evitando los tiempos de inactividad, a la vez que facilita tareas comunes de administrador del sistema. PM2 también permite gestionar el registro de aplicaciones, la supervisión y la agrupación en clúster.

Las principales características de PM2:

- Ver el estado de distintas apps
- Capacidad de monitoreo de memoria y CPU de los procesos
- Manejo de logs
- Balanceo de carga.

Tras configurarlo todo, en la siguiente interfaz se puede observar el estado de la aplicación:

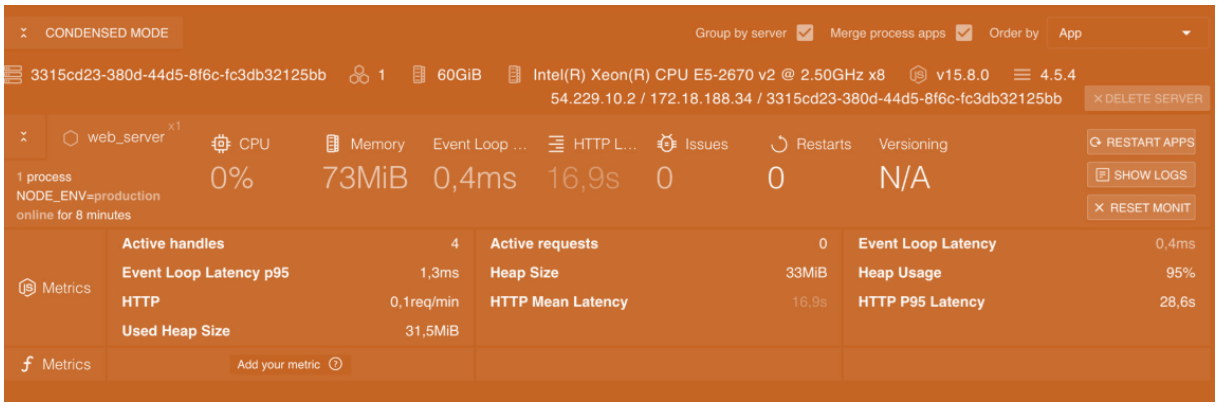


Ilustración 36: Interfaz de PM2 en la que se puede ver el servidor back-end de GAIA

Tras realizar las mejoras explicadas, se han añadido y actualizado todas las evidencias a mostrar en la presentación ante la Secretaría de Estado para la Agenda Digital (Ministerio de Economía y Empresa). En el momento de escribir estas líneas, dicha inspección está aún pendiente de ser ejecutada.

2.5.4 Colaboración diaria en Tentu

El desarrollo principal en el que participa ISEA es Tentu. Tentu, la revista electrónica personalizada fue lanzada al público del País Vasco en diciembre de 2018. Debido a ello se registraron alrededor de 2000 usuarios por lo que el equipo de desarrollo recibió muchísimo feedback[24] con posibles mejoras para la aplicación.

Aunque mi labor no ha consistido en implementar ninguna funcionalidad de la aplicación web en concreto, sí que he sido participe del desarrollo, aportando mi punto de vista en cada una de las decisiones creativas que debía tomar el equipo semanalmente.

Entre las nuevas funcionalidades implementadas, son destacables las siguientes:

- Rediseño de la interfaz, destacando los contenidos, entre otras cosas
- Sección de comentarios en cada uno de los contenidos
- Posibilidad de ver las noticias más visitadas
- Nueva sección “Tablón de anuncios” en la que los usuarios pueden subir sus propios contenidos

El fichero XD con los prototipos de las interfaces está disponible en el anexo.

Ilustración 37: Interfaz principal de Tentu

[24] Término utilizado para referirse a la opinión de los usuarios respecto a una aplicación.

3 Resultados

Tras describir detalladamente el desarrollo del proyecto, en este apartado se hará un análisis de los resultados obtenidos.

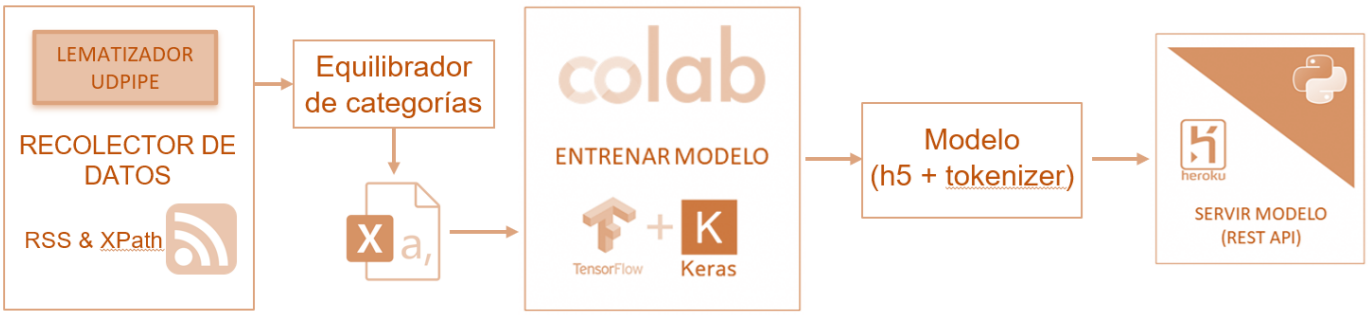


Ilustración 38: Sistema implementado para la clasificación textual

Gracias a este diseño, se pueden incorporar temáticas nuevas y actuales en el clasificador de forma mucho más rápida y sencilla, ya que solamente es necesario reunir diferentes fuentes RSS en el recolector.

Tras ello, el sistema implementado se encarga de seleccionar y ponderar las palabras más significativas, reduciendo así la labor humana en trabajos repetitivos y con poco valor añadido.

Además de ello, el porcentaje de acierto respecto al sistema de clasificación de UZEI ha aumentado de un 75% a un 83% con la posibilidad de aumentar ese valor en un futuro a medida que se recopilen más artículos mediante el recopilador, por lo que tiene margen de mejora.

En cuanto al recolector de textos, este se ocupa de realizar parte del preproceso de los datos, reduciendo así el trabajo del analista de datos, ya que solamente debe entrenar el modelo y ajustar parámetros con el fin de obtener el mejor resultado.

Desde la perspectiva del análisis de datos, se han dejado preparados unos Jupyter Notebooks en Google Colab con todos los algoritmos implementados con el fin de que el desarrollador no deba preocuparse de preparar ningún entorno.

Observando la arquitectura implementada para conseguir un modelo de filtrado de contenidos:

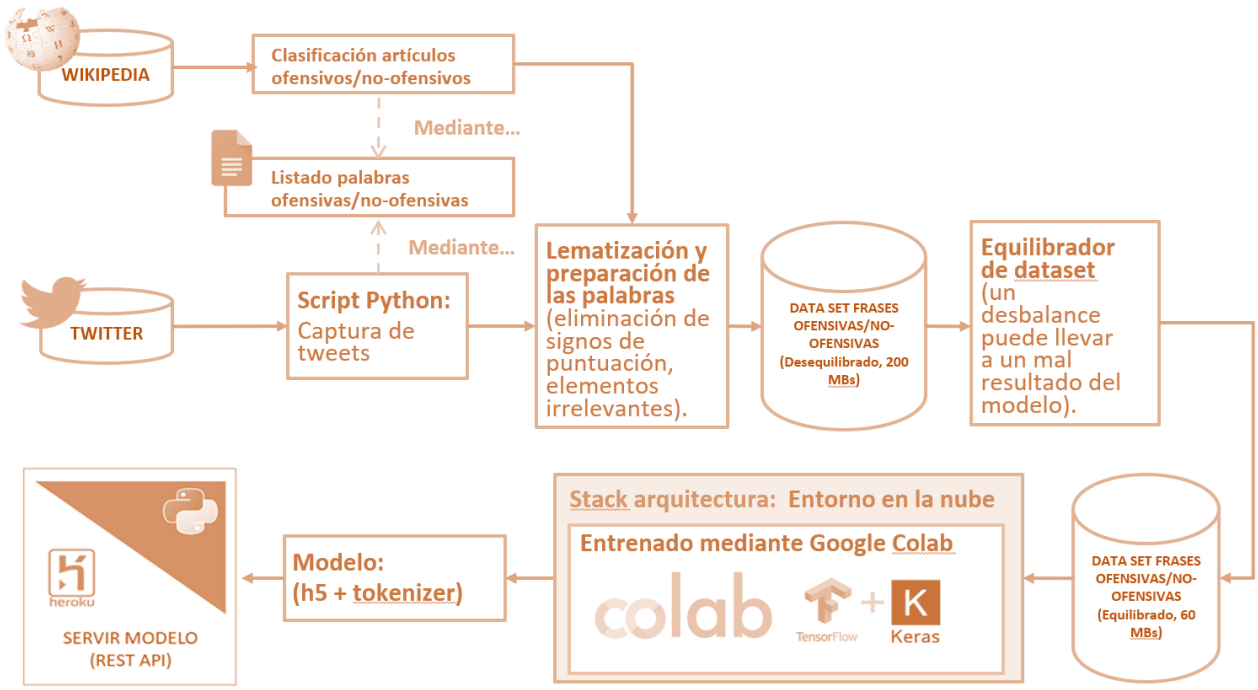


Ilustración 39: Sistema implementado para el filtrado de textos

El resultado de este diseño es un modelo de filtrado con capacidad de acierto del 86% que visto el contexto en el que tiene que trabajar en el apartado 2.3.4 se puede decir que es un porcentaje adecuado.

Este sistema va a permitir a sistemas como Tentu implementar un sistema de comentarios en el que sus usuarios podrán debatir sobre sus contenidos sin redactar nada ofensivo o tóxico.

Además, al igual que en el clasificador de han dejado listos diferentes notebooks en la nube con el fin de incluir en el modelo posibles nuevos conjuntos de palabras tóxicas. Con el fin de obtener estos nuevos conjuntos tóxicos, también se han dejado listos diferentes scripts con los que poder obtener tanto el conjunto como su contexto por Twitter.

Para terminar, se ha participado en desarrollos adicionales que han aportado valor y provocado que el conocimiento adquirido sea mayor. A destacar la migración del proveedor del servidor de EvidenceBox, donde se ha encontrado y aplicado una alternativa más potente y económica, aumentando así su capacidad computacional.

Todo lo realizado ha conllevado obtener conocimientos sobre diferentes ámbitos como pueden ser:

- Desarrollo de código en el lenguaje Python y Javascript
- Gestión de entornos mediante PIP y Conda
- Gestión y control de versiones con GitLab
- Gestión, desarrollo de pipelines de integración, entrega y despliegue continúo mediante GitLab CI/CD
- Gestión de testeos con PYTEST
- Entrenamiento de modelos en entornos en la nube, haciendo uso de GPUs alquiladas
- Proveedores en la nube como pueden ser Google, Amazon o Heroku.

4 Memoria económica del proyecto

En el siguiente apartado se detallará el coste de las herramientas utilizadas para la realización del proyecto.

4.1 Coste instrumental y equipamiento

En este apartado se detalla el coste de los equipos y elementos utilizados para el desarrollo del proyecto.

Elemento	**Cantidad**	**Coste**
MacBook Pro (13 inch with Touch Bar, 2018)	1	1600€
Otros accesorios para el ordenador (HUB, adaptadores USB)	1	50€
Otros accesorios como cuadernos, bolígrafos...	1	20€

Tabla 8: Coste instrumental y equipamiento

4.2 Coste personal

En este apartado se detallan los costes del personal implicado en el proyecto.

Personal	**Coste por hora**	**Horas dedicadas**	**Coste**
Ander Bolumburu	4,36€	1050 horas	4578€

Tabla 9: Coste personal

4.3 Coste total

Teniendo en cuenta los anteriores apartados, el coste total de este proyecto ha sido de 6258€.

5 Conclusiones y líneas futuras

En este apartado, se comentarán las conclusiones que se pueden deducir del proyecto.

5.1 Conclusiones técnicas

Teniendo en cuenta el apartado sobre los objetivos del proyecto y sobre el estado y planificación de este, se puede concluir que el proyecto se ha realizado correctamente y el desarrollo de tareas ha sido el correcto, habiéndose cumplido los objetivos establecidos en la planificación inicial, la cual figura en el apartado 1.4 de este mismo documento.

En cuanto a las competencias a desarrollar por el alumno, se estima que se han cumplido correctamente:

- **CTFM01:** Redacta correctamente el informe del trabajo final de máster y presenta y defiende con claridad el resultado de dicho trabajo.
    - La redacción del presente informe, junto con la presentación del proyecto como defensa de este.

- **CTFM02:** El alumno/la alumna se ha integrado óptimamente en la empresa, colaborando con el resto de las personas de su entorno, y ha demostrado alto nivel de resolución y autonomía en el desarrollo del Trabajo Fin de Máster.
    - La integración en la empresa se ha llevado a cabo satisfactoriamente, creando tanto relaciones profesionales como personales. El grupo de trabajo ha sido capaz de desarrollar el producto con el fin de presentar una solución atractiva para el cliente.

- **M1N102:** Diseñar, desarrollar e implementar técnicas de preprocesamiento y modelado de datos para predecir, clasificar y agrupar los mismos, siendo capaz de interpretar y validar los modelos creados para la extracción del conocimiento.
    - Se ha diseñado e implementado un proceso de preprocesamiento adaptado al problema que ocupa (tratamiento de textos) con fuentes contrastadas. Además, se han investigado y validado varios algoritmos con el fin de buscar el que mejor resultado proporcionaba.

- **M1N301:** Utilizar herramientas informáticas del análisis de datos en nuevos campos de aplicación para resolver problemas complejos y realizar proyectos de ingeniería teniendo en cuenta el contexto comercial e industrial.
    - Para llevar a cabo el proyecto se ha hecho uso de varias herramientas informáticas del análisis de datos como pueden ser SciKit – Learn y Keras. Por otro lado, se ha sabido aprovechar la experiencia en otros ámbitos (programación, computación en la nube) para sacar ventaja y llevar a cabo un proyecto más enriquecedor en cuanto a conocimiento.

- **M1N303:** Utilizar herramientas informáticas para el desarrollo de aplicaciones y operaciones (DevOps), tanto a nivel local como en la nube, para resolver problemas complejos y realizar proyectos de ingeniería teniendo en cuenta el contexto comercial e industrial.
    - Se han utilizado varias herramientas informáticas para el desarrollo de aplicaciones tanto en local (Python, principalmente) como en la nube (Heroku, AppEngine). Además, se ha querido dar énfasis a la parte académica del proyecto y se han aprendido herramientas desconocidas como pueden ser AWS ECR y ECS, entre otras.

5.2 Conclusiones metodológicas

Dentro de los métodos empleados, se entiende que es el correcto, dado el buen estado que presenta el proyecto y que el tiempo estimado ha sido el suficiente para llevar a cabo todas las tareas planteadas. El conocimiento de la organización de la empresa también ha sido muy útil.

Se ha hecho uso del tablero que proporciona Gitlab, para planificar las tareas que se han tenido que ir haciendo en cada una de las fases/milestones.

Además, el hecho de que el equipo de desarrollo tenga experiencia previa en el desarrollo con diferentes herramientas como Flask, Node o React ha agilizado el desarrollo del proyecto, permitiendo avanzar en otras áreas como pueden ser el desarrollo del modelo capaz de clasificar diferentes temáticas o el rediseño de una de las aplicaciones de la organización como es Tentu.

5.3 Conclusiones personales

El desarrollo del proyecto resultó ser llevadero, ya que contaba con dos años de experiencia previas en la empresa. Es por ello por lo que conocía tanto la organización interna del código, como la forma de trabajar de la empresa.

A pesar de llevar dos años en la empresa, el desarrollo del proyecto conllevaba trabajar en ámbitos de la informática que no había trabajado en ISEA. Es por ello por lo que al comienzo resultó un poco complicado ya que mis conocimientos sobre la inteligencia artificial y el desarrollo y despliegue de aplicaciones en la nube estaban limitados a los recibidos en la Universidad.

Por suerte, el proyecto me ha permitido profundizar y afianzar conceptos en este ámbito, dando valor a la sabiduría adquirida en el máster, ya que me ha hecho ver que son tecnologías que están en auge hoy en día, siendo muy utilizadas en varios ámbitos o entornos.

Desde la perspectiva del recolector de textos se han cumplido todos los objetivos propuestos al inicio del desarrollo como pueden ser que el propio recolector hiciera parte del preproceso de los datos, entre otros. Y no solo se han conseguido, sino que se han analizado diferentes opciones y tras comprobar resultados se ha elegido la mejor. Todo esto conlleva un

proceso de aprendizaje y experimentación, que visto el marco académico del proyecto, aporta mucho valor y empaque al proyecto en general.

En cuanto al análisis de datos, se han analizado e implementado una gran variedad de algoritmos y tras escoger la opción que mejor resultado proporcionaba, se ha desarrollado un software que posibilite la consulta a los clientes. Asimismo, se ha hecho uso de buenas prácticas de programación como son la integración y el despliegue continúo entre otras, con el fin de realizar un software de calidad.

Además, se le debe dar valor a que todo el entrenamiento ha sido realizado en un entorno en la nube, haciendo que el entrenamiento se pueda ejecutar desde cualquier lugar.

Por otro lado, desde un principio se ha atendido a los consejos y aportes del equipo en cuanto a la organización de la estructura de la aplicación se refería. Si en un futuro fuera preciso añadir una nueva funcionalidad, ello no resultará complicado. Además, una buena organización de la aplicación conduce a una mejor comprensión por parte de otros futuros desarrolladores.

En cuanto a la experiencia trabajando, he de reconocer que los conocimientos que he adquirido durante este tiempo han sido, globalmente, coincidentes con los que he desarrollado durante el máster. Si bien, obviamente, en la Universidad se trabaja de forma tutelada, mientras que en un entorno real resulta prioritaria la efectividad y la conclusión de los objetivos.

Complementariamente, el proyecto me ha permitido investigar acerca de diferentes tipos de neuronas con las que poder formar inteligencias artificiales, como pueden ser las convolucionales con Word Embedding, que han presentado unos resultados sorprendentes y pueden llegar a superar al modelo con mayor porcentaje de acierto.

Por otra parte, destacaría el conocimiento adquirido en el desarrollo de aplicaciones alojadas en AWS ECS de Amazon. Esta clase de plataformas, en las que el desarrollador se olvida de mantener un servidor, además de resultar muy útiles, están en auge y creo que el hecho de conocerlas aportará mucho valor al desarrollador del futuro.

Finalmente, las practicas han supuesto una gran fuente de conocimiento, ya que antes de comenzarlas mi conocimiento se limitaba al adquirido en la universidad. Ahora, una vez sumergido en el desarrollo, no solo he descubierto la gran variedad de herramientas existentes en el mercado, si no que he aprendido varias en profundidad. No he llegado a un dominio absoluto, pero si dispongo de un conocimiento suficiente para desenvolverme con el fin de poder crear y desplegar aplicaciones como el recolector o el modelo de inteligencia artificial.

5.4 Aporte de valor a un producto existente

Mediante las prácticas y conocimientos obtenidos por ellas se ha conseguido aportar valor a un producto existente como es GAIA. Tras el desarrollo, se pueden añadir nuevas categorías de forma muy rápida y sencilla al sistema, sin necesidad de haber labor humana de por medio. Dicha labor humana retrasaba un poco la adición de nuevas temáticas al sistema ya que se debían seleccionar y ponderar cada una de las palabras correspondientes a la nueva temática.

5.5 Líneas futuras

Como líneas futuras de este proyecto, inicialmente se plantea atraer al mayor número de clientes posibles al uso de los servicios.

Desde el punto de vista del proyecto GAIA, se podría investigar sobre la idea de que cada uno de los clientes disponga de un modelo de predicción totalmente personalizado con los temas que la persona usuaria le interesen. Ya que en el supuesto caso en el que solamente interese diferenciar textos relacionados con, por ejemplo, el coronavirus y el deporte y el hecho de que el resto de las temáticas vayan incluidas en el modelo, reste eficacia al resultado final.

Asimismo, sería interesante que la persona usuaria pudiera comprobar a la hora de realizar la contratación, la eficacia del modelo con las temáticas seleccionadas, evitando así malentendidos, ya que el porcentaje de acierto variaría en función de las temáticas seleccionadas.

De este modo, se conseguiría que el modelo solamente se centrase en las temáticas que a la persona usuaria le interesasen, siendo el producto mucho más atractivo de cara al cliente final.

Igualmente, con el fin de añadir mayor exactitud al modelo, se podrían crear una especie de submodelos, con subcategorías más concretas como puede ser en el caso de los deportes. En el modelo principal estarían incluidos artículos de todos los deportes de forma general, pero sería este submodelo el que sería capaz de diferenciar de que deporte concreto se trata. De esta forma, la persona usuaria tendría un grado más de exactitud.

Otra posible mejora, como se ha podido observar en el apartado de resultados, las redes neuronales convolucionales juntadas con Word Embeddings, conseguían un resultado que se aproximaba a las ANNs. Teniendo en cuenta que estas Embeddings solamente cubrían un 75% del vocabulario, una mejora puede pasar por conseguir un Embedding que cubra una mayor parte del vocabulario.

Además, cabría la posibilidad de aumentar la compatibilidad con otros idiomas haciendo uso de diferentes APIs de traducción y utilizando el inglés como puente entre los idiomas.

Por otro lado, desde el punto de vista de las buenas prácticas del desarrollador, sería adecuada la diferenciación de entornos completa en los desarrollos de la empresa. Actualmente únicamente existen diferentes entornos en las aplicaciones y estos comparten la misma versión del back-end y bases de datos, lo que ha provocado en numerosas ocasiones la obligación de actualizar la versión en producción debido a incompatibilidades.

En cuanto a los modelos, además de la personalización comentada al inicio de la sección, sería adecuado mejorar los datos, como se ha indicado en la reflexión del apartado de resultados, sobre todo del modelo de filtrado de textos, identificando en diferentes escalas las palabras bien/mal sonantes.

Asimismo, en lugar de tener que actualizar los componentes (modelo, encoder y tokenizador) del servicio de forma manual cada vez que se obtiene un mejor resultado, estaría bien automatizar un proceso en el que el sistema comprobase si con los nuevos datos se puede obtener un modelo con mejor resultado al actual.

Desde el punto de vista del despliegue automático de estos servicios, estaría bien duplicar su despliegue en diferentes entornos, e incluso conseguir un despliegue zero-down-time, para ello sería necesario adquirir conocimiento de Kubernetes. Al ser un producto que aún no está pensado para poner en producción, no se ha querido invertir tiempo en ello, pero sí que estaría bien tenerlo en cuenta para el futuro.

En cuanto al desarrollo de la APP para smartphones de Tentu, sería adecuado automatizar tanto la integración y el despliegue de la aplicación, como la publicación en sus respectivas tiendas, restando así trabajo a los desarrolladores.

Por otra parte, la gestión de las fuentes a consultar por parte del recolector de artículos puede resultar complicada para la gente sin conocimientos de formatos JSON/ programación. Es por ello por lo que resultaría adecuado el desarrollo de una pequeña interfaz web mediante la cual poder gestionar el listado de una forma sencilla e intuitiva.

Para finalizar, en cuanto a posibles mejoras del recolector, se podría investigar sobre lematizadores que vayan incluidas en la aplicación de forma local y no depender de servicios de terceros desplegados en la nube. Además, con el fin de ahorrar costes, sería interesante investigar sobre Serverless Lambda, ya que este tipo de servicios únicamente se pagan por uso, con lo que la empresa se ahorraría el hecho de tener una máquina constantemente encendida.

5.6 Posible desarrollo de una nueva empresa

En el momento de escribir estas líneas, ISEA se encarga del desarrollo de varios softwares:

- Tentu, la revista electrónica personalizada junto a todos sus servidores.
- GAIA, servicio de clasificación y filtrado de textos en la nube.
- EvidenceBox, servicio de fedatario para gestiones online.

Es por ello por lo que no es descartable la creación de una nueva empresa de desarrollo de software, abstraída del ámbito innovador en el que se mueve ISEA.

Esto posibilitaría la atracción de nuevos clientes y el completo enfoque de la nueva empresa a estos productos.

5.7 Agradecimientos

A mis padres y hermano, que siempre han estado ahí para ayudarme en los momentos más difíciles.

A mis tutores y compañeros de trabajo.

6 Valoración personal

La realización del proyecto en ISEA S. Coop. ha sido muy satisfactoria ya que he aprendido a trabajar en una empresa con un equipo reducido para el desarrollo. Algo no muy habitual ya que normalmente las empresas dedicadas al desarrollo de software cuentan con grupos grandes.

En mi caso, he estado trabajando en esta empresa desde septiembre de 2018 y durante este tiempo he aprendido a desarrollar software desde otro punto de vista.

Gestionar las *builds* automáticas, implementar testeos de calidad u otros elementos de la ingeniería del software, eran procesos que solamente conocía en un ámbito puramente académico. Este proyecto me ha permitido ampliar mis conocimientos sobre todo ello e implementarlo en el desarrollo, aportando así código de mayor calidad.

Adquirir un mayor conocimiento en herramientas relacionadas con el desarrollo de aplicaciones y la inteligencia artificial como Docker, Keras, Heroku o AWS ha sido un gran reto. Han existido momentos de frustración debido al desconocimiento acerca de la herramienta, pero al final por medio de la experiencia ha sido posible llevar a cabo el desarrollo.

Además, el hecho de haber aprendido a desplegar contenedores en una herramienta como AWS ECS supone un plus importante a mis conocimientos. Ya que además de ser algo totalmente innovador, debido a que el desarrollador se olvida completamente del mantenimiento del servidor, es una tendencia en auge. Diferentes proveedores como la propia Amazon, Google o Microsoft están invirtiendo auténticas millonadas en este tipo de plataformas, con el fin de mejorarlas y aportar un valor diferencial a los clientes. Esta competitividad entre ellas lo único que hace es beneficiar al cliente, en este caso nosotros, los desarrolladores.

Por parte de la Universidad, mi tutor Alain Pérez a quien agradezco su ayuda, siempre atento y dispuesto a solventar las posibles dudas que pudiera tener.

Por otro lado, me he dado cuenta de que el hecho de trabajar aporta un conocimiento añadido, ya que te permite aplicar los conocimientos adquiridos en el máster en entornos reales.

Asimismo, estimo que la realización del proyecto en una empresa ha sido una experiencia interesante y me ha aportado un gran conocimiento. En efecto, gracias a este proyecto he desarrollado nuevas capacidades que, dentro de proyectos llevados a cabo en la universidad, como los POPBL, no hubiese sido capaz de desarrollar.

No me gustaría finalizar la redacción de este informe sin agradecer a Aitor Orobengoa y todo su equipo en ISEA por la paciencia y comprensión tenida con mi delicada situación familiar. Siempre han estado ahí para apoyarme en los momentos más duros. Una vez más, gracias de corazón.

7 Anexos

En el siguiente apartado se podrán encontrar los anexos citados durante el documento:

ANEXO A: Gantt

En este fichero se incluye el diagrama de Gantt mostrado a una mayor resolución.

ANEXO B: Fichero XD

Archivo con formato XD que contiene los prototipos de la interfaz de Tentu en la que el equipo de diseño ha estado trabajando.

8 Bibliografía

[1] «Kyocera Document Solutions,» [En línea]. Available: https://www.kyoceradocumentsolutions.es/es/smarter-workspaces/insights-hub/articles/diferencia-entre-datos-estructurados-y-no-estructurados.html.

[2] A. F.-S. y. E. A. Sánchez-Pérez, «Open data, big data: ¿hacia dónde nos dirigimos?,» 2012.

[3] D. I. M. V. R. A. A. Marcelo Errecalde, «Tópicos avanzados en categorización de textos,» 2009.

[4] M. I. C. G. R. Aida Slavic, «El desarrollo de la Clasifi cación Decimal Universal: 1992-2008 y más allá,» 2009.

[5] A. M. &. J. R. Pérez-Agüera, «SKOS: Simple Knowledge Organisation for the Web,» 2007.

[6] B. P. J. H. Ralf Steinberger, «Cross-lingual Document Similarity Calculation,» 2002.

[7] R. Troncy, «Bringing the IPTC News Architecture into the,» 2008.

[8] «Chakray,» 2017 Enero 27. [En línea]. Available: https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/.

[9] M. S. y. A. Shatter, «DIRECTIVA 2013/37/UE DEL PARLAMENTO EUROPEO Y DEL CONSEJO,» 2013.

[10] M. A. M. Antonín, Tecnologías del lenguaje, Barcelona: UOC, 2003.

[11] J. Wijffels, «UDPipe Webpage,» 10 12 2020. [En línea]. Available: https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-annotation.html.

[12] S. T. &. P. R. Gupta, «Python as a Tool for Web Server Application Development,» Delhi, 2014.

[13] N. M. a. R. Schneeman, Heroku: Up and Running, O' Reilly Media, 2013.

[14] Diginmotion, «About Amazon Web Services,» 2010.

[15] A. R. A. T. Mehryar Mohri, Foundations of Machine Learning, 2018.

[16] Y. B. a. A. Ian Goodfellow, Deep Learning, 2016.

[17] R. L. d. Mántaras, «BBVA Open Mind,» [En línea]. Available: https://www.bbvaopenmind.com/articulos/el-futuro-de-la-ia-hacia-inteligencias-artificiales-realmente-inteligentes/.

[18] V. I. Group. [En línea]. Available: https://www.ucavila.es/images/files/GuiaAcademica/18-19/titPropios/IA/Informe_IA.pdf.

[19] M. Dhande, «Geospatial World,» 7 3 2020. [En línea]. Available: https://www.geospatialworld.net/blogs/difference-between-ai%EF%BB%BF-machine-learning-and-deep-learning/.

[20] R. González, «El test de Turing: Dos mitos, un dogma,» 2007.

[21] E. A. J. B. L. B. U. C. R. G. J. M. G. B. L. M. M. M. S. Antonio Moreno, Aprendizaje Automático, Barcelona, 1994.

[22] APD, «APD,» 4 3 2019. [En línea]. Available: https://www.apd.es/que-es-machine-learning/.

[23] A. L. a. M. Wiener, «Classification and Regression,» 2002.

[24] A. N. Rich Caruana, «An empirical comparison of supervised learning algorithms,» 2006.

[25] H. B. Barlow, Unsupervised Learning, 1989.

[26] Z. Ghahramani, Unsupervised Learning, 2004.

[27] Y. B. a. G. H. Yann LeCun, «Deep Learning,» 2015.

[28] T. A. J. F. C. S. Ilija Ilievski, «Efficient Hyperparameter Optimization for Deep Learning,» Singapore, 2017.

[29] L. D. V. S. J. C. C. D. A. M. P. C. T. R. R. L. M. F. J. D. S. S. P. v. P. C. M. Y. J. J. P. C. X. T. L. S. Thomas Wolf, «State-of-the-art Natural Language Processing,» Brooklyn, 2020.

[30] S. L. Team, «SciKit Learn Library Documentation,» [En línea]. Available: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html.

[31] J. Ramos, «Using TF-IDF to Determine Word Relevance in Document Queries,» Piscataway.

[32] S. L. D. Team, «SciKit Learn guide: SVM,» [En línea]. Available: https://scikit-learn.org/stable/modules/svm.html.

[33] S. L. D. Guide, «SciLearn Guide: Multinomial Naive Bayes,» [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.

[34] S. L. D. Team, «SciKit Learn guide: Logistic Regression,» [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[35] S. L. D. Team, «SciKit Learn guide: Decision Tree Classifier,» [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[36] J. Hopfield, «Artificial neural networks,» 1988.

[37] S. E. Dreyfus, «Artificial Neural Networks, Back Propagation,» Berkeley, 1990.

[38] G. H. A. K. I. S. R. S. Nitish Srivastava, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» Toronto, 2014.

[39] S. G. M. G. V. V. Pinkesh Badjatiya, «Deep Learning for Hate Speech Detection in Tweets,» Hyderabad, India, 2017.

[40] S. M. Robert Bamler, «Dynamic Word Embeddings,» 2017.

[41] R. S. C. D. M. Jeffrey Pennington, «GloVe: Global Vectors for Word Representation,» Stanford, 2014.

[42] N. Ketkar, Introduction to Keras, 2017.

[43] H. L. &. J. Song, Introduction to convolutional neural network using Keras, 2019.

[44] F. Chollet, Deep Learning with Python, ANAYA, 2017.

[45] J. Moolayil, Learn Keras for Deep Neural Networks, Apress, 2019.

[46] R. S. a. H. N. Martin Sundermeyer, «LSTM Neural Networks for Language Modeling,» 2012.

[47] H. K. K. &. V. J. R. Krishnaiah, «A comparative study of K-Means, DBSCAN and OPTICS,» 2016.

[48] N. V. J. J. V. Aristidis Likas, «The global k-means clustering algorithm,» 2003.

[49] J. S. M. E. H.-P. K. X. X. Erich Schubert, «DBSCAN Revisited: Why and How You Should (Still) Use DBSCAN,» 2017.

[50] S. U. R. S. F. S. S. Kamran Khan, «DBSCAN: Past, Present and Future,» 2014.

[51] E. Bisong, Building Machine Learning, Deep Learning Models on Google Cloud Platform, Berkeley, 2019.

[52] J. Brownlee, «Machine Learning Mastery,» 14 Agosto 2020. [En línea]. Available: https://machinelearningmastery.com/difference-test-validation-datasets/.

[53] P. G. B. a. G.-C. Vosniakos, «Optimising feedforward artificial neural network architecture,» 2007.

[54] Y. S. a. B. W. Wah, «Global Optimization for Neural Network Training,» 1996.

[55] F. H. Vera-Rivera, «Método de automatización del despliegue continuo en la nube para la implementación de microservicios,» 2018.

[56] P. M. A. B. N. M. S. P. F. C. Alicia Salamon, «La Integración Continua Aplicada en el Desarrollo de Software en el Ámbito Científico – Técnico,» Buenos Aires, 2014.

[57] Luis Alberto Iñiguez Sánchez, «Arquitectura tecnológica para la entrega continua de software con despliegue en contenedores,» Cuenca (Ecuador).

[58] W. Wu, «React Native vs Flutter, cross-platform mobile,» 2018.

[59] M. S. &. S. Gozi, «REGLAMENTO (UE) No 910/2014 DEL PARLAMENTO EUROPEO Y DEL CONSEJO,» 2014.

[60] S. Ifrah, Deploy Containers on AWS, 2019.

[61] D. Gonzalez, Developing Microservices with Node.js, Packt Publishing Ltd., 2016.