UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño, desarrollo e implementación de un algoritmo de análisis de sentimientos en publicaciones de Twitter

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Telecomunicación

AUTOR/A: Pons Moro, David

Tutor/a: Guerola Navarro, Vicente

CURSO ACADÉMICO: 2022/2023

Resumen

Actualmente, la sociedad se encuentra en una era en la que gran parte de la comunicación se realiza mediante el uso de las redes sociales. En algunas de estas, como puede ser el caso de Twitter, los usuarios muestran mediante publicaciones su opinión o vivencias sobre diversos temas, muchas veces sobre temas de actualidad o ideológicos. Esto hace, entre otras cosas, que esta red social sea una fuente para tener en cuenta cuando se busca saber cuáles son esas últimas noticias que han sacudido la sociedad o qué tendencias hay últimamente. Analizar las publicaciones que estos usuarios hacen respecto a temas/tendencias puede ayudar a comprender mejor que respuesta está teniendo una parte de la sociedad a ello. Es por eso por lo que utilizar este tipo de técnicas puede marcar drásticamente la diferencia en ámbitos que abarcan desde la creación de estrategias de marketing a la política. El propósito de este proyecto es precisamente crear una herramienta que sea capaz de analizar las publicaciones realizadas en Twitter respecto a un tema (frase o palabra clave), pudiendo detectar mediante inteligencia artificial qué sentimientos predominan en cada publicación, ya sean positivos, negativos o neutros. Esta información se correlacionará con datos demográficos de los usuarios, cuyas publicaciones han sido utilizadas en el estudio, para hacer más sencilla la detección de ciertos patrones, que pueden ser utilizados a favor del usuario de esta herramienta.

Resum

Currently, society is in an era in which a large part of communication is carried out through the use of social networks. In some of these, such as Twitter, users show through publications their opinion or experiences on various topics, often on current or ideological issues. This makes, among other things, this social network a source to take into account when seeking to know what are the latest news that have shaken society or what trends have recently emerged. Analyzing the posts that these users make regarding topics/trends can help to better understand what response a part of society is having to it. This is why using these types of techniques can make a drastic difference in areas ranging from the creation of marketing strategies to politics. The purpose of this project is precisely to create a tool that is capable of analyzing the publications made on Twitter regarding a topic (phrase or keyword), being able to detect through artificial intelligence what feelings predominate in each publication, whether positive, negative or neutral. This information will be correlated with demographic data of the users whose publications have been used in the study to make it easier to detect certain patterns, which can be used in favor of the user of this tool.

Abstract

Actualment, la societat es troba en una era en la qual gran part de la comunicació es realitza mitjançant l'ús de les xarxes socials. En algunes d'aquestes, com pot ser el cas de Twitter, els usuaris mostren mitjançant publicacions la seua opinió o vivències sobre diversos temes, moltes vegades sobre temes d'actualitat o ideològics. Això fa, entre altres coses, que aquesta xarxa social siga una font per a tindre en compte quan es busca saber quines són aqueixes últimes notícies que han sacsejat la societat o quines tendències hi ha últimament. Analitzar les publicacions que aquests usuaris fan respecte a temes/tendències pot ajudar a comprendre millor que resposta està tenint una part

de la societat a això. És per això que utilitzar aquest tipus de tècniques pot marcar dràsticament la diferència en àmbits que abasten des de la creació d'estratègies de màrqueting a la política. El propòsit d'aquest projecte és precisament crear un algoritme que siga capaç d'analitzar les publicacions realitzades en Twitter respecte a un tema (frase o paraula clau), podent detectar mitjançant intel·ligència artificial quins sentiments predominen en cada publicació, ja siguen positius, negatius o neutres. Aquesta informació es correlacionarà amb dades demogràfiques dels usuaris, les publicacions dels quals han sigut utilitzades en l'estudi, per a fer més senzilla la detecció d'uns certs patrons, que poden ser utilitzats a favor de l'usuari d'aquest programa.

A mis padres y mi hermana, por ser ese apoyo incansable.

A mi tutor Vicente, por todo ese valor aportado.

A mi mismo, por haber conseguido finalizar esta etapa

Índice general

Índice de figuras

Listado de siglas empleadas

AGI Artificial General Intelligence.

ANI Artificial Narrow Intelligence.

API Application programming interface.

ASI Artificial Super Intelligence.

BERT Bidirectional Encoder Representations from Transformers.

GPT Generative Pre-trained Transformer.

HTTP Hypertext Transfer Protocol.

IA Inteligencia Artificial.

JSON JavaScript Object Notation.

LSTM Long Short-Term Memory.

NLP Natural Language Procesing.

REST Representational State Transfer.

RNN Recurrent Neural Network.

RoBERTa Robusted Optimized Bidirectional Encoder Representations from Transformers.

SMTP Simple Mail Transfer Protocol.

SOAP Simple Object Access Protocol.

TF-IDF Term Frequency-Inverse Document Frequency.

XML Extensible Markup Language.

Parte I

Memoria

Capítulo 1

Introducción

1.1. Contexto y justificación del trabajo

Estos últimos años se han vivido grandes avances respecto a las IA con las apariciones de algoritmos de generación de texto como ChatGPT o de generación de imágenes como MidJourney. Estas inteligencias artificiales plantean un cambio en la sociedad actual que empieza a tener repercusión en entornos como el laboral o la educación entre otros. Esta nueva etapa plantea una serie de oportunidades que pueden resultar beneficionas en múltiples sectores así como de preocupaciones como puede ser el decremento de la cultura del esfuerzo en los jóvenes, la pérdida del pensamiento critico, la perdida de empleos por poder ser remplazados por inteligencias artificiales....

Centrándonos en las ventajas que estas nuevas inteligencias, una de ellas es el análisis y clasificación de emociones en personas, ya sea mediante el análisis facial, el análisis de textos escritos o el reconocimiento por voz. Esta rama de la inteligencia artificial cuenta con múltiples aplicaciones como en el de el muestreo de opiniones en campañas de marketing o políticas.

Es en este contexto de crecimiento, ligado a mi interés por la psicología, me decido a crear un algoritmo capaz de clasificar los textos en diferentes emociones para usos como los comentados anteriormente. De esta forma me introduciré en este nuevo mundo, trataré de comprender los aspectos técnicos y crearé una herramienta con la que poder comprender que sentimientos predominan en ciertos temas de la actualidad.

1.2. Conceptos Básicos

1.2.1. Emociones

Adicionalmente nos encontramos en una época en la que cada vez se le da mayor importancia al componente emocional de las personas. En este caso nos vamos a centrar en el la importancia que tiene a la hora de formar opinión así como en el ámbito del marketing donde está altamente presente en estos días.

Las 7 emociones de Paul Ekman

La teoría de las emociones de Paul Ekman es una de las más influyentes y reconocidas en el campo

de la psicología y con repercusión en los ámbitos académico, científico y social. En esta se propone que existen seis emociones básicas que son universales y se expresan a través de gestos faciales específicos, estas son: ira, asco, miedo, alegría, tristeza y sorpresa [1].

Ekman comenzó sus investigaciones en la década de 1960, inspirado por la hipótesis de Charles Darwin en el que se argumentaba que las emociones tienen un origen biológico y no uno cultural. Para comprobarlo, viajó a diferentes lugares del mundo y observó las expresiones faciales de personas de distintas etnias, culturas y lenguas. Descubrió que las seis emociones básicas se manifestaban de forma similar en todos los grupos humanos, lo que demostraba su carácter universal.

En 2003 Ekman realizó una publicación llamada .El rostro de las emociones"donde describe con detalle cada una de las emociones básicas así como sus causas, sus efectos y sus expresiones faciales asociadas. También explica cómo reconocer las microexpresiones y cómo entrenar la habilidad de leer el rostro de los demás. Según Ekman, esta habilidad puede ser útil para mejorar la comunicación, la empatía y la detección de mentiras.

El marketing emocional:

A partir de la década de 1980, las empresas comenzaron a darse cuenta de que no era suficiente poner en valor solo las características técnicas de sus productos. Necesitaban conectarse emocionalmente con los consumidores para destacar en un mercado repleto de opciones. Es en 1982 que Tom Peters y Robert H. Waterman Jr. escriben un libro titulado ´´En búsqueda de la excelencia" donde se analizan aquellas empresas más exitosas del momento y resaltando la importancia de poner en valor el componente emocional. Las principales características de esta nueva tendencia se ven reflejados a continuación [2]:

- Conexión emocional: Se busca poder establecer una conexión profunda y duradera con los consumidores, buscando que estos simpaticen con los valores, emociones y experiencias compartidas que esta marca brinda. Esto muchas veces se consigue a través del *Storytelling*. Un ejemplo podrían ser los anuncios de la lotería de navidad o de *Coca-Cola*
- Experiencia del cliente: Se enfoca en brindar una experiencia excepcional al cliente, donde las emociones positivas tienen un papel fundamental.
- Identidad de marca: Se busca desarrollar una identidad de marca sólida y auténtica, con principios bien definidos, que resuene emocionalmente con el público objetivo.
- Comportamiento del consumidor: Se estudian las emociones y los comportamientos de los consumidores para comprender mejor sus motivaciones y necesidades emocionales así como que repercusión ha tenido en el mercado cierto producto o campaña.

1.3. Objetivos del trabajo

En este proyecto trato de crear una herramienta capaz de recoger información de twitter sobre un tema en especifico o de un conjunto de datos y analizar que sentimientos predominan en estos discursos. Además también se busca implementar dos gráficos que representen tanto el numero total de tweets por emoción como la variación de esas emociones a lo largo del tiempo, pudiendo localizar así picos o tendencias interesantes a analizar.

Para ello he definido una serie de subtareas claves para poder abordar este proyecto:

- **Informarse** sobre el análisis de sentimientos mediante inteligencia artificial.
- **Realizar test** con diferentes modelos de *Sentiment Analysis*
- **Realizar tests** con diferentes opciones de obtención de *tweets*.
- **Unificar** en un solo algoritmo.
- **Implementar gráficos** para correlacionar información
- **Documentar** el proyecto realizado.

1.4. Planificación del trabajo

A la hora de planificar temporalmente el proyecto, se ha tenido en cuenta los objetivos definidos en el apartado anterior y mediante una gráfica de Gantt se han definido unos intervalos temporales aproximados para poder planificar correctamente el proyecto. En la figura 1.1 se pueden ver representados dichos intervalos:

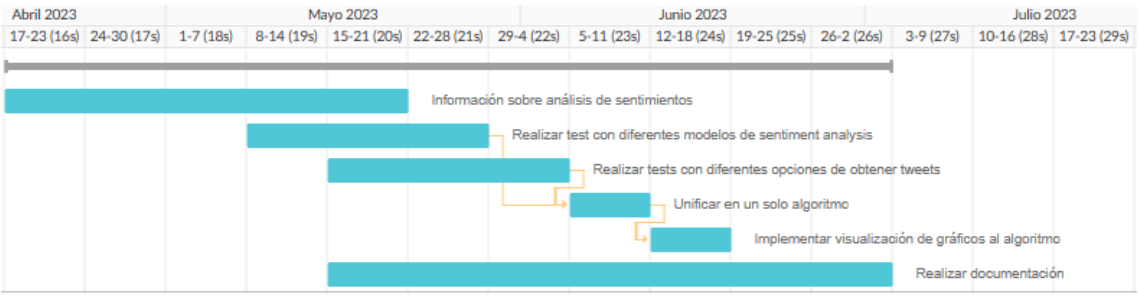


Figura 1.1: Diagrama de Gannt

Además, para el desarrollo de subtareas y análisis de múltiples errores, se ha seguido una metodología *Kanban*. Para ello, se ha hecho uso de la aplicación *Notion* en la cual se ha creado una base de datos con aquellas tareas a realizar y se a creado una vista en modo tablero la cual se puede visualizar en la figura 1.2. De esta forma se puede visualizar de forma simultanea aquellas tareas no finalizadas pudiendo distinguirse en que proceso del desarrollo están ubicadas. En este caso las fases seleccionadas han sido: *Not started*, *In progress*, *Pending*, *Done*

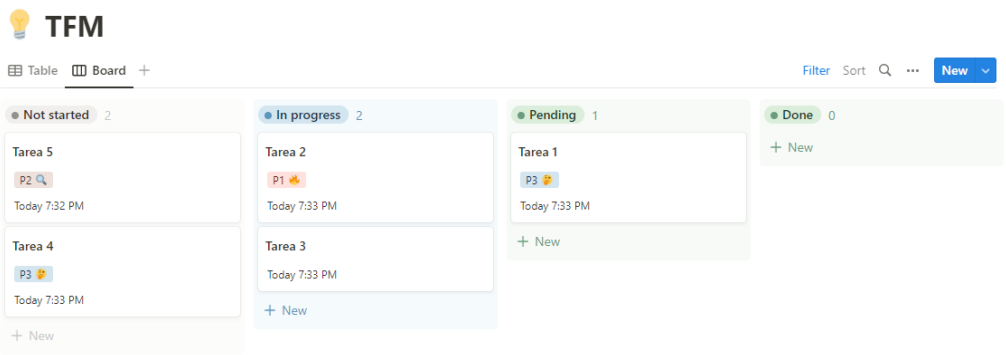


Figura 1.2: Metodologia Kanban en Notion

Capítulo 2

Marco teórico

2.1. Inteligencia Artificial (IA)

La inteligencia artificial (IA) es una nueva rama de de la informática que ha tenido un crecimiento exponencial desde la mitad del siglo pasado llegando a ser . Este crecimiento ha sido posible gracias a los avances en hardware, que brindan mayor capacidad de procesamiento, así como el desarrollo de algoritmos de IA más complejos cada vez para ámbitos más diversos. El fin de esta es poder imitar el comportamiento humano, pudiendo aprender, realizar tareas y adaptarse a nuevas situaciones de forma autónoma.

Los orígenes de la inteligencia artificial se remontan a 1950 donde el considerado padre de las matemáticas, Alan Turing, realizó la publicación de *Computing Machinery and Intelligence* [3]. En esta, se abordaba la cuestión sobre si las maquinas tenían facultades para poder pensar por si mismas.

Para contestarse a dicha pregunta Alan propone la llamada *"Prueba de Turing"*. Esta consiste en dos actores clave: un ordenador y un examinador. El examinador mantendría una conversación mediante ordenador con una supuesta persona que no estaría visible para el. Es entonces donde el examinador debe decidir si según la conversación mantenida el otro locutor es una persona o una maquina. De llegar a la situación en la que una maquina consigue engañar al examinador, esta maquina habría pasado la prueba.

2.1.1. Aplicaciones

Con sistemas como esto se consiguen múltiples aplicaciones en diversos campos entre ellos se puede destacar:

- **La medicina:** Dentro de la medicina, la *IA* está siendo de utilidad en diferentes tareas. Una de ellas es utilizar utilizar la *IA* para que se especialice en la detección de ciertos patrones que previamente ha aprendido. De esta forma se podrá realizar un diagnostico de una manera más certera y precoz. Esto puede ser de gran utilidad en enfermedades de rápida evolución, como algunas enfermedades degenerativas o algunos tipos de cáncer. Ya que en este tipo de enfermedades una rápida detección puede ser crucial.

Un ejemplo de aplicación es la detección de cáncer de mama. En casos como este, herramientas especializadas en detectar este tipo de anomalías pueden llegar a ser capaces de eliminar cierta subjetividad a la hora interpretar las radiografías. De esta forma se consigue mejorar la precisión de los radiólogos. [4]

Figura 2.1: Imágenes de una mamografia
[5]

- **La educación:** Esta es una de las areas sobre la que ha habido un reciente impacto debido a la aparición de *chatbots* como es el caso de *Chatgpt*. Estos son capaces de entender, resolver y explicar gran parte de las tareas a los que estudiantes se someten continuamente, desde redacción de documentación hasta análisis y resolución de problemas relativamente complejos. Las inteligencias artificiales pueden ser de gran ayuda tanto para los estudiantes como para los educadores con funcionalidades como estas:[6]

  - Realización de tutorías más personalizadas.
  - Edición de programas de estudios basados en el ritmo de cada uno de los estudiantes.
  - Actualización de conocimientos de los docentes, ayudándoles a implementar nuevas metodologías.
  - Ayudar a predecir abandono escolar y poder proponer soluciones.

Figura 2.2: Digitalización de las aulas
[7]

- **Ingeniería:** En el ámbito de la ingeniería también han habido grandes avances promovidos por la inteligencia artificial. Uno de los casos más llamativos son los siguientes [8]:

  - **Mantenimiento:** A la hora de monitorizar el deterioro de la maquinaria en una fabrica, se ha empezado a optar por sensorizar componentes. Con esto se puede se crea un sistema de supervisión capaz de obtener en tiempo real información del estado de los componentes y poder predecir su vida útil. De esta forma se podrán substituir aquellos componentes antes de que fallen y afecten al correcto funcionamiento del sistema.
  - **Automatización:** Actualmente, cada vez están surgiendo más robot denominados *pick and place*. Estos son capaces de mover/colocar objetos desordenados mediante el uso de cámaras con reconocimiento en tiempo real y evitando en gran medida errores. Algunas de las tareas que realizan guardan relación con la logística, el embalaje o el ensamblaje de piezas entre otros.

Figura 2.3: Maquinas ”pick and place”
[9]

2.1.2. Clasificaciones

La IA tiene diversas formas de clasificarse según criterios como la capacidad de aprendizaje o la complejidad de los problemas que pueden resolver. Si nos centramos en la primera opción, nos encontramos con tres categorías diferentes: [10]

- **ANI:** Inteligencias con memoria limitada que están destinadas a cumplir objetivos concretos, siendo poco útiles en otras tareas poco relacionadas con para las que son programadas. En esta categoría es donde actualmente están todas las *IA* conocidas hasta la fecha.
- **AGI:** Inteligencias que conseguirían igualar las capacidades de un humano gracias a sus capacidades cognitivas y de aprendizaje. Esto permitirá que una única *IA* pueda realizar tareas de ámbitos completamente diferentes al mismo nivel que un humano. Tareas como resolver un problema matemático, ”pintar un cuadro”, dialogar.... Actualmente ningún modelo a llegado a este estado aunque hay empresas investigando para alcanzarlo en futuras versiones. Un ejemplo de ellas es la empresa *OpenIA*.
- **ASI:** Con este tipo de inteligencias se busca entre otras cosas conseguir la retención ilimitada de información o la auto-consciencia. Con habilidades como estas podrían dar lugar a *IAs* que conseguirían sobrepasar las cualidades de los seres humanos.

Respecto a la segunda distinción, desde 1980 ha ido evolucionando la inteligencia artificial dando a lugar a capas más complejas de esta llamadas *Machine Learning* y *Deep Learning*. Estas serán explicadas a continuación y se pueden ver representadas de forma gráfica a continuación 2.4

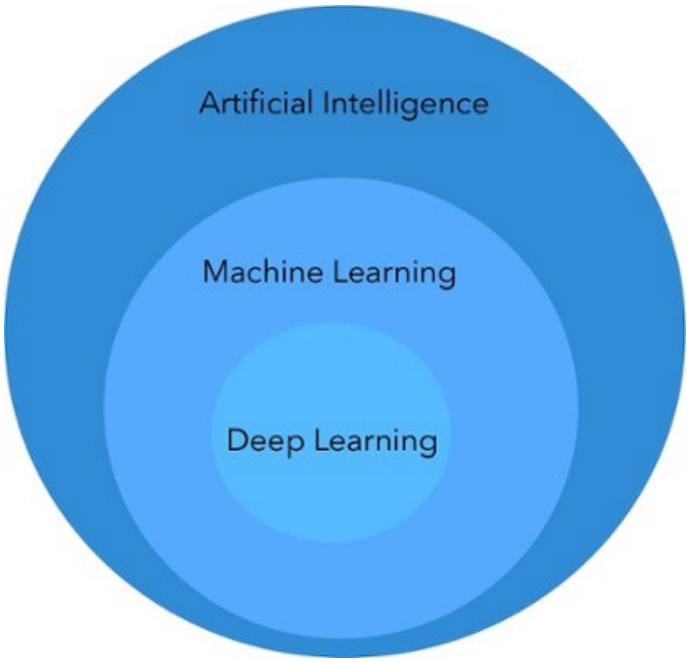


Figura 2.4: Capas dentro de la IA

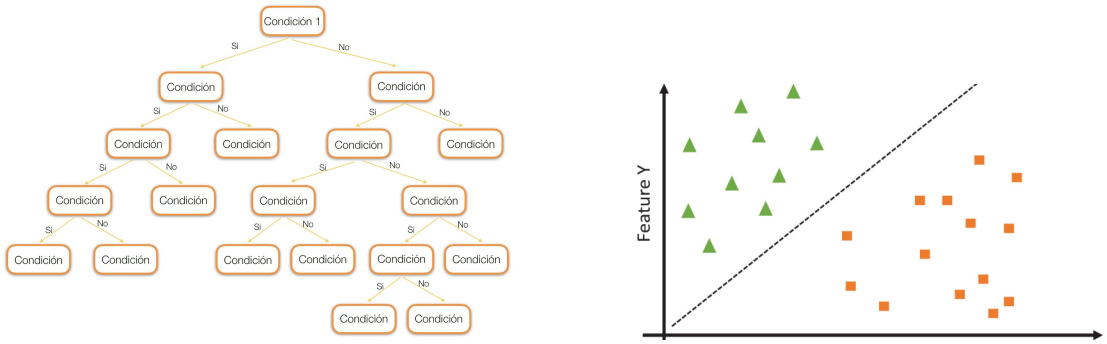
2.1.3. Machine Learning

Esta técnica busca dotar a las maquinas de la capacidad de aprendizaje, para ello se usan diferentes técnicas como:

- Aprendizaje **supervisado**.
- Aprendizaje **no supervisado**.
- Aprendizaje **reforzado**.

Con ello se busca poder entrenar a las *IA* y mejorar su aprendizaje en cada iteración del proceso descartando comportamientos erróneos y favoreciendo los correctos. Una forma de saber que comportamientos son los correctos o incorrectos es mediante el uso del *ground truth*. [1]

Si se habla de modelos especializados en la clasificación, algunos de estos son los arboles de decisión [11] o *support vector machine* [12]. Un ejemplo gráfico de su funcionamiento puede ser visualizado en la figura 2.5



(a) Árbol de decisión (b) Support Vector Machine (SVM)

Figura 2.5: Modelos de clasificación en Machine Learning

En el primer caso, teniendo en cuenta las características de la imagen/texto a analizar, se toman decisiones hasta llegar a una conclusión. Por ejemplo, si se quisiese diferenciar entre fotos de una farola o una pelota, una de las características distintivas a analizar podría ser su redondez y en el árbol de decisión esa condición podría clasificar en dos grupos.

Respecto a la segunda opción se basa en asignar a cada eje del plano dos características distintivas y representar en el plano un punto dependiendo cuanto cumplan cara característica. A continuación se intentará clasificar trazando un vector que de forma iterada tratará de buscar la pendiente y origen para tener una distancia máxima e igual en ambos lados respecto a la nube de puntos.

En este caso se ha explicado esta técnica con dos grupos y dos dimensiones aunque puede añadirse complejidad mediante el uso de mas dimensiones (características a diferenciar) o mediante la distinción de más de dos grupos.

[1] conjunto de datos etiquetados creados de forma manual con el objetivo de que una *IA* pueda utilizarlos como referencia para optimizar su aprendizaje

Durante el paso del tiempo, el *Machine Learning* ha implementado nuevas técnicas, entre ellas se podría destacar el *clustering*, modelos mejorados capaces de resolver un abanico más amplio de tareas [13], así como también el aprendizaje profundo (*DeepLearning*)

Respecto a como se consigue extraer las características necesarias para realizar el análisis, este se realiza de forma manual con técnicas como:*One-hot Encode* [14] (por cada características solo el elemento más apto del conjunto obtiene el valor de 1 y el resto de elementos obtienen el valor de 0), *TF-IDF* [15](que muestra cuanto de relevante es un elemento respecto al conjunto en cada característica)...

2.1.4. DeepLearning

Este se basa en añadir un mayor número de capas a las redes neuronales, siendo estas cada vez mas profundas. Con esto, junto con un mayor número de datos de entrada a los que se utilizarían en *Machine Learning*, las redes neuronales consiguen aprender de forma totalmente autónoma siendo capaces de extraer características clave de los datos proporcionados por si mismas para su posterior utilización 2.6. Esto abre la puerta a nuevas funcionalidades como pueden ser el reconocimiento de voz, texto o imágenes, la generación de textos,imágenes o musica, la clasificación de elementos de forma más autónoma y precisa o la automatización de sistemas robóticos.

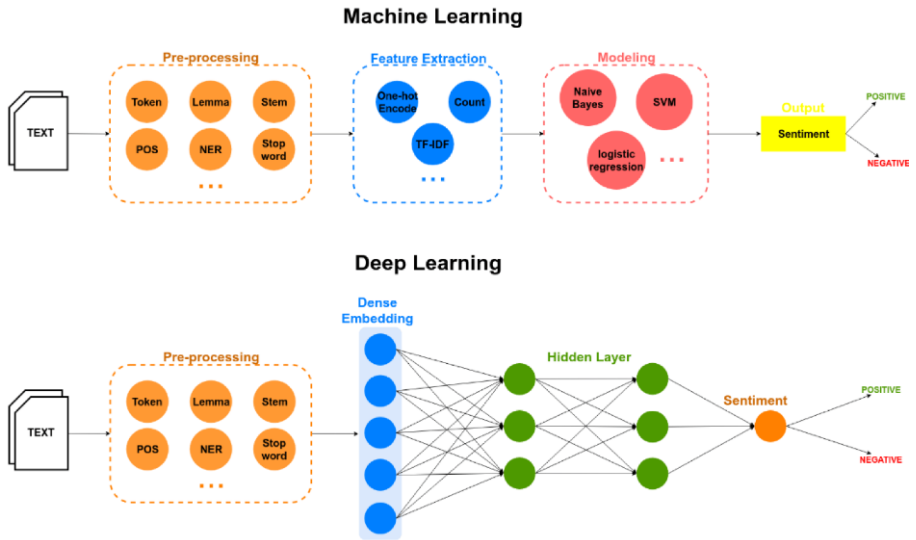


Figura 2.6: Diferencia entre Machine Learning y Deep Learning
[16]

2.2. Natural Language Processing (NLP)

El *Natural Language Procesing* (NLP) es la rama del *DeepLearning* centrada en estudiar la comunicación humana. Con esto se busca crear algoritmos capaces de comprender a un usuario ya sea de forma hablada o escrita y que puedan realizar diversas tareas. Estas pueden ser la generación de texto asemejándose lo máximo posible a la forma de escritura humana, clasificación de

los datos de entrada en la temática a la que pertenece, clasificación de los datos de entrada en sentimientos/emociones o completar frases con aquellas palabras más probables entre otras muchas aplicaciones.

El NLP combina diferentes campos de estudio como la lingüística computacional, modelos matemáticos estadísticos así como técnicas de *Machine Learning* y *Deep Learning*. Esto permite procesar el significado, la relevancia, el sentimiento y la intención del lenguaje natural [17].

2.2.1. Historia del NLP

La historia del NLP se remonta a los años 50 del siglo pasado, cuando se realizaron los primeros intentos de traducción automática entre idiomas. Uno de los proyectos pioneros fue el sistema Georgetown-IBM, que traducía textos del ruso al inglés con un vocabulario limitado y unas reglas gramaticales simples [18].

En los años 80 y 90, el NLP experimentó un cambio de paradigma hacia el uso de métodos estadísticos y probabilísticos, que aprovechaban la disponibilidad de grandes corpus de textos anotados y etiquetados lingüísticamente. Estos métodos permitieron crear modelos matemáticos que aprendían las regularidades y las variaciones del lenguaje a partir de los datos, sin necesidad de reglas explícitas. Algunas de las aplicaciones que surgieron en esta época fueron los sistemas de reconocimiento y síntesis de voz [19], los sistemas de recuperación y extracción de información [20], y los sistemas de resumen automático [21].

En el siglo XXI, el NLP ha experimentado una nueva revolución gracias al avance de las técnicas de aprendizaje profundo (deep learning), que consisten en el uso de redes neuronales artificiales con múltiples capas de procesamiento. Estas redes son capaces de extraer características abstractas y complejas del lenguaje a partir de grandes cantidades de datos no estructurados, y de generar textos coherentes y fluidos. Algunas de las aplicaciones más recientes y avanzadas del NLP basado en deep learning son los sistemas de traducción automática neuronal [22], los sistemas de generación de texto condicionada con modelos como GPT [23], y los sistemas de comprensión lectora [24].

2.2.2. Aplicaciones del NLP

Entre sus aplicaciones podemos destacar:

- **Asistentes virtuales (Comprensión de texto)**: Entre ellas se encontrarían asistentes disponibles en nuestros móviles o ordenadores como Siri, Cortana o Alexa. La función de estos es comprender el mensaje del usuario, ya sea escrito o mediante el uso de la voz y realizar una tarea. Estas tareas pueden ser desde llamar a ciertos contactos, realizar búsquedas en internet o interactuar con dispositivos del hogar conectados a internet.
- **Análisis de sentimientos**: Se encarga de analizar un texto dado, de longitud variable y ser capaz de detectar que sentimientos hay implícitos. La clasificación generalmente se hace en *Positivo*, *Negativo* o *Neutro* aunque hay algoritmos capaces de obtener mayor precisión. Estos son los llamados algoritmos de análisis de emociones los cuales pueden discernir entre *Ira*, *Asco*, *Miedo*, *Alegría*, *Neutro*, *Tristeza* y *Sorpresa* generalmente.

  Esta técnica se utiliza entre otras cosas en atención al cliente. Para ello se hace uso de algoritmos como este y técnicas que permiten transcribir de audio a texto para permitir monitorizar

en todo momento como evolucionan los sentimientos de un cliente durante una llamada.

- **Traducción automática**: Esta es otra de las funcionalidades que presenta el NLP. Hay múltiples herramientas que disponen de inteligencia artificial a la hora de realizar sus traducciones. De esta forma la traducción es mucho más natural que con las técnicas anteriores. Un ejemplo de esto es DeepL.
- **Generación de texto**: La generación de texto tiene muchas aplicaciones prácticas, como la escritura de resúmenes, artículos, correos electrónicos o poemas. Para realizar dicha generación de texto, se utilizan modelos computacionales y diferentes sets de datos utilizados para entrenar al modelo. Con esto tratan de imitar la estructura o escritura. Algunos de los modelos más avanzados de generación de texto, como GPT (modelo que está detrás del conocido chatGPT) son capaces de producir textos con coherencia, siendo capaces de adaptar su discurso, de realizarlo en diferentes idiomas y dominios.
- **Transcripción**: Otra de sus aplicaciones consiste en ser capaz de convertir la información de un archivo de audio a texto, pudiendo crear subtitulos de videos incluso en diferentes idiomas.

2.2.3. Redes neuronales utilizadas

- **RNN:** Las redes neuronales recurrentes (RNN por sus siglas en inglés) son una de las redes más básicas utilizadas en el NLP. Estas se caracterizan por prestar atención a la temporalidad de una determinada secuencia lo que las diferencia de otras redes básicas como las CNN (utilizadas para el procesamiento de imágenes por su atención espacial). El funcionamiento de una RNN se puede observar de forma esquemática en el figura 2.7. Este consiste en realimentar el sistema para que en la próxima salida del sistema tenga en cuenta la salida anterior, de esta forma las frases generadas obtendrán mayor coherencia. [25]

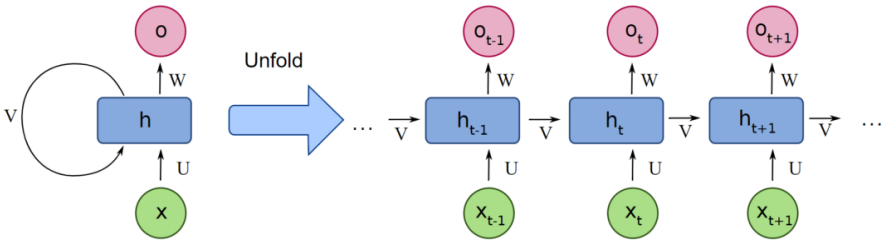


Figura 2.7: Esquema de una RNN
[16]

Esta red tiene una serie de limitaciones, entre ellas la falta de memoria. Si por ejemplo se quiere que la red complete la frase “Me encanta la comida asiática, voy a ir a cenar a....”, esta probablemente lo hará de forma errónea. Esto se debe a que la influencia de los estados anteriores va decreciendo de forma lineal en cada iteración independientemente de la importancia de la palabra. Es por eso que a la hora de completar la frase, la influencia de “comida asiatica” será muy baja.

- **LSTM :** Las redes *long short-term memory* buscan suplir los problemas de memoria de las RNN bajo un principio diferente. Este consiste en no dar la misma importancia a todas las

palabras de una frase ya que no todas serán igual de relevantes. A la hora de implementar esto, se añade una nueva realimentación llamada "celda de estado" (representada como C en la figura 2.8) en la que se añadirán aquellas palabras clave que influenciarán en futuras iteraciones. [26]

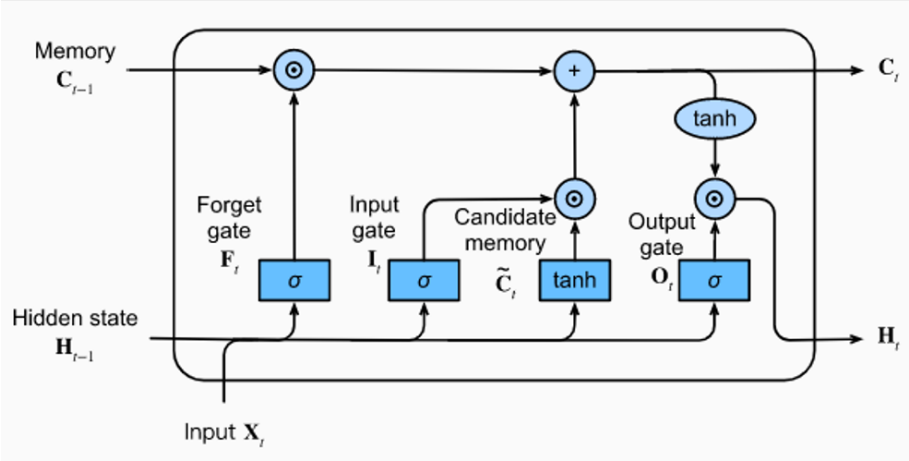


Figura 2.8: Esquema de una red LSTM
[26]

Como se ha podido ver en la figura, esta red consta de tres *gates* en las que se interaccionará con la celda de estado:

- **Forget Gate**: permite decidir si la información almacenada en la celda de estado de la iteración anterior es importante ser conservada para futuras iteraciones. Para ello toma el estado oculto anterior y la entrada actual (H y X), los transforma y los lleva a la función de activación sigmoidal. La salida de esta tendrá valores cercanos a 0 o 1 y será multiplicada por la celda del vector C. De esta forma se eliminará o mantendrá la información previamente guardada.
- **Update Gate**: En esta parte del proceso, se crea un vector de valores candidatos (*Candidate memory* a ser añadidos a la celda de estado. A continuación se filtran los valores realizando una multiplicación punto a punto con el vector generado en el *Input gate*.El resultado de esta multiplicación será sumará a la celda de estado actualizando de esta forma su información
- **Output Gate**: A la hora de generar el estado oculto de salida, este no es más que una versión filtrada de lo añadido previamente en la celda de estado. Para ello, se usa la *Output Gate* para definir que posiciones de la celda de estado pasarán a formar parte de la salida. Seguidamente se realizará una multiplicación punto a punto que filtrada aquellos datos seleccionados.

- **Transformers:** La red *Transformers* fue descrita por primera vez el año 2017 en una publicación realizada por investigadores de Google llamada *Attention is all you need* [27]. En ella se hablaba del avance que suponía añadir bloques de atención los cuales estarían encargados de investigar las relaciones que guardaban los elementos de una secuencia escrita. De esta forma se obtenían mejores que con las redes anteriores. La topología de esta red es la siguiente:

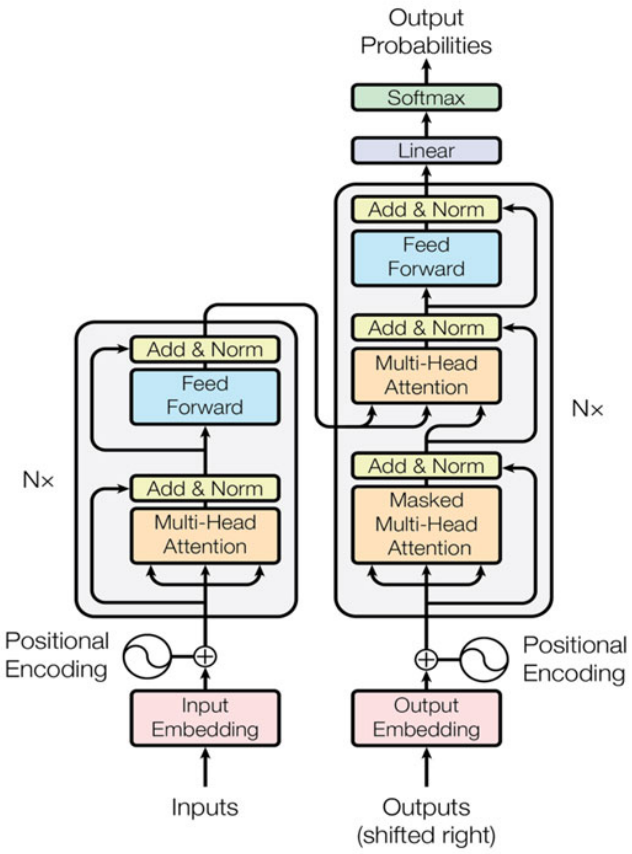


Figura 2.9: Representación esquemática de la red Transformer

La "atencion" en esta red se consigue gracias a los módulos *Multi-Head Attention*. A continuación se va a explicar el funcionamiento de este modulo teniendo en cuenta los módulos anteriores de la entrada.

Primeramente el módulo *Input Embedding* codifica cada elemento de la secuencia de caracteres en tokens/vectores de forma que pueda ser entendida por la red y seguidamente, mediante el *Positional Encoding*, se le añade información posicional respecto a la secuencia mediante una suma de vectores.

Es ahora cuando el módulo de atención calculará la relación que guarda cada palabra con el resto de ellas. Para calcular la relación, dado que cada palabra es un vector, se realiza una multiplicación escalar de todos los elementos entre ellos dando lugar a la matriz de atención 2.10. Estas multiplicaciones serán convertidas posteriormente a probabilidades y finalmente, a la salida de este modulo, se obtendrá un vector por cada palabra. Este nos dará información sobre cuanta relación guarda con cada una de las palabras restantes.

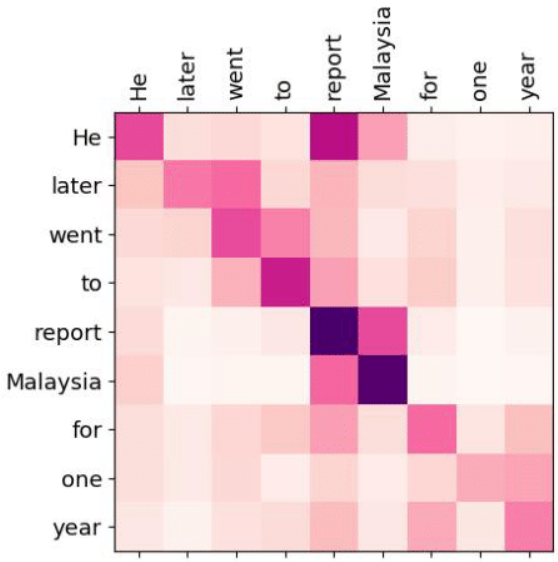


Figura 2.10: Matriz de atención
[28]

2.2.4. Modelos de NLP

Como se ha explicado anteriormente, el NLP cuenta con diversas aplicaciones. Es por eso que existen una gran variedad de diferentes modelos de inteligencia artificial en los que varia su topología o los datos utilizados para el entrenamiento. Algunos modelos, como los que se muestran a continuación, tienen en cuenta la topología *transformers* comentada en el subapartado anterior.

- **BERT** (Bidirectional Encoder Representations from *Transformers*)
- **RoBERTa** (Robusted Optimized Bidirectional Encorder Representations from *Transformers*)
- **GPT** (Generative Pre-trained *Transformer*)

2.3. Application Programming Interface (API)

Una interfaz de programación de aplicaciones (API) es un *software* intermediario que permite la comunicación entre un cliente (solicitante) y un servidor.

En otras palabras, permite que dos o más aplicaciones se comuniquen entre sí y compartan datos y funcionalidades sin necesidad de conocer mas detalles de los que la propia API pide para su funcionamiento. De forma muy esquematizada se puede observar la utilidad de dicha herramienta en la figura 2.11

Como se ha comentado anteriormente, para el uso de la API es necesario tener en cuenta como interactuar con ella. Es por eso que se necesita conocer bien la documentación en la que se especifica entre otras muchas cosas que tipo de protocolo se utiliza, de lo cual hablaremos más en el siguiente

Figura 2.11: Representación esquemática de una API

apartado, que estructura seguir para realizar las peticiones o que respuesta y con que información se debe esperar en cada petición [29].

2.3.1. SOAP y REST

Como bien se comentaba al final del apartado anterior, conocer el protocolo de las APIs es vital a la hora de entender como se conecta a internet, como comparte la información o que requiere para su manutención. Es por eso que a continuación se hablará de dos términos muy ligados a las APIs los cuales son SOAP y REST.

Por un lado **SOAP** utiliza el formato XML en las respuestas del servidor y las peticiones de sus usuarios pueden ser realizadas tanto por HTTP como por SMTP. Esto genera una ventaja a la hora de trabajar con esta api ya que permite la interacción con aplicaciones de clientes diferentes

Por el otro lado **REST** es el tipo de *API* más conocido, este utiliza el protocolo *HTTP* y mas concretamente el método *CRED* (*PUT*, *GET*, *POST* y *DELETE*) para realizar las peticiones. Por otro lado, los formatos de archivo esperados en las respuesta son de tipo JSON o XML. [30]

Dentro del grupo *REST* se hace una distinción entre *RESTful* y *RESTless*, dicha distinción viene marcada por si una *API* cumple estrictamente con los criterios REST (Restful) o no completamente con todos (Restless). A continuación se enumeran algunos de los requisitos REST a seguir: [31].

- **Arquitectura cliente-servidor** Lo que quiere decir que dicha arquitectura se basa en un cliente el cual realiza de forma activa las peticiones y un servidor el cual administra los recursos y responde a las solicitudes.

Figura 2.12: Gráfica cliente-servidor
[31]

- **Sistema sin estado** Esto quiere decir que cada una de las peticiones son independientes entre si, ya que no se se almacena en el servidor el estado de las interacciones de cada usuario conectado.

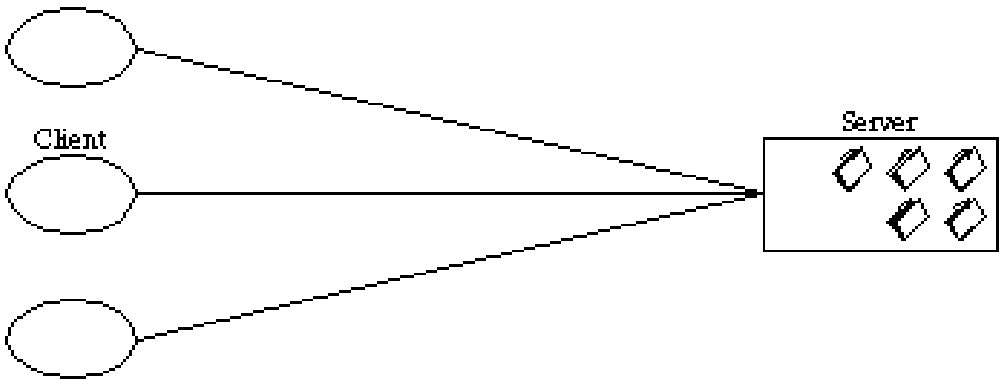


Figura 2.13: Gráfica sistema sin estado
[31]

- **Capacidad de almacenamiento en caché** Tambien se busca el intentar minimizar ciertas peticiones recurrentes con el servidor, es por esto que la información enviada por el servidor será etiquetada como *cacheable* o *non-cacheable*. Por eso mismo si el cliente recibe una respuesta *cacheable*, esta podrá ser almacenada por el cliente para reusarse posteriormente si es necesario.

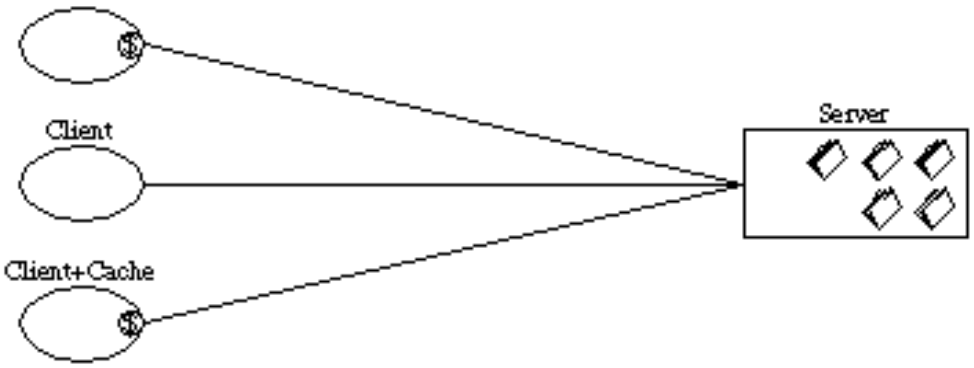


Figura 2.14: Gráfica cliente-servidor
[31]

- **Sistema en capas** La arquitectura de cliente-servidor puede contar con capas intermedias que le añadan nuevas funcionalidades al servicio, ejemplos de ello son añadir sistemas de caché compartido, opción a balanceo de cargas etc...

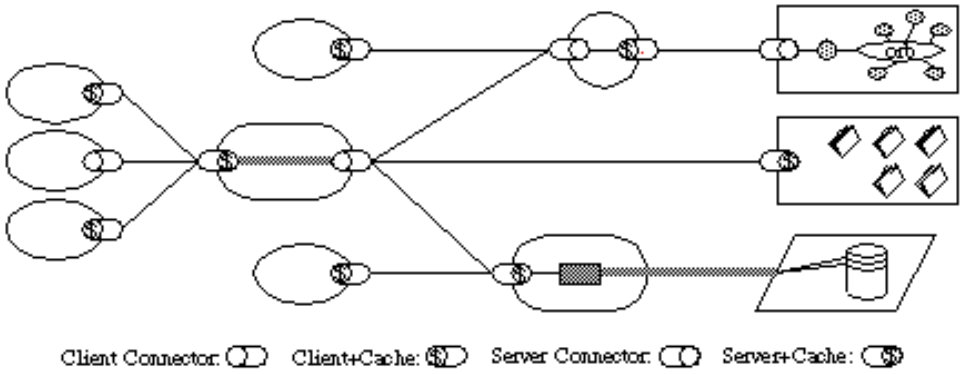


Figura 2.15: Gráfica arquitectura a capas
[31]

2.3.2. Autenticación

Generalmente para hacer uso de las APIs se suelen necesitar una serie de códigos o *tokens* que se solicitarán desde la pagina oficial de dicha API y que se deberán almacenar en el propio archivo de programación, en variables de entorno o en archivos de configuración. Estos códigos/*tokens* son necesarios para autenticar al desarrollador a la hora de que este interactúe con la API. Los principales motivos por los que se implementa autenticación a las API's son:

- **Seguridad:** Permite identificar al desarrollador y comprobar que es un usuario autorizado, de esta manera no se pone en peligro los datos o recursos a los cuales se accede a través de la API
- **Control de acceso:** Permite limitar el acceso a ciertos recursos dependiendo de que rol se se le haya asignado a cada usuario, de esta forma se evita que todos los que tienen acceso a la API tengan acceso a la totalidad de su información si no es necesario.
- **Facturación:** En algunas API's el proveedor cobra por el uso de su API por lo que es necesario tener identificado cada usuario que hace uso de dicha aplicación. Alguna de las formas de tarificación son monitorizando el uso realizado por el usuario, por lo que se le cobra por lo utilizado, o bien creando diferentes planes con diferentes limitaciones, en estas cada usuario podrá elegir que opción se adapta más a sus necesidades.

2.3.3. Ejemplos de uso

Aunque las aplicaciones de estas son mucho mas extensas. A continuación se expondrá algunos de los ejemplos de uso más comunes de las API's.[32]

- **Integración de redes sociales:** Redes sociales como Twitter, Facebook o Instagram cuentan con APIs para aquellos usuarios que decidan desarrollar programas que interactuen con dichas plataformas. Algunos de las aplicaciones de estas son programas de publicación automática, herramientas de análisis de datos, bots para la respuesta automática de mensajes directos....

- **Integración de servicios de mapas:** Los servicios de mapas son aplicaciones que ofrecen información geográfica, como puede ser la ubicación, la distancia o la ruta a seguir a la hora de realizar un trayecto. Algunos de estos servicios de mapas más conocidos son Google Maps, Bing Maps y OpenStreetMap entre otros. Estas plataformas cuentan con APIs que permiten a los desarrolladores integrar sus mapas y sus funcionalidades en otras aplicaciones. Entre ellos, la API de Google Maps permite hacer todo tipo de cosas con mapas en las páginas web, como mostrar la ubicación actual del usuario, calcular la distancia entre dos puntos, trazar rutas y mostrar información adicional sobre los lugares.

- **Procesamiento de pagos:** Algunos de los servicios de procesamiento de pagos más utilizados actualmente son PayPal, Stripe y Mercado Pago[33]. Es aqui donde de nuevo las API's dan la oportunidad de integrar sus sistemas de pago en otras aplicaciones. Algunos de losejemplos que estas funcionalidades aportan son:

    - la API de PayPal permite aceptar pagos con tarjeta o con cuenta de PayPal en una tienda online.
    - la API de Stripe permite crear y gestionar suscripciones y facturas en una plataforma SaaS.
    - la API de Mercado Pago permite procesar pagos con diferentes medios en una aplicación móvil.

Capítulo 3

Material empleado

3.1. ChatGPT 1000 Daily Tweets

Cuenta con aproximadamente 43000 tweets Este conjunto de datos [34] en formato CSV cuenta con aproximadamente 43000 tweets que guardan relación con “ChatGPT”, “GPT3” o “GPT4”, además cada dia se actualiza añadiendo unos 1000 nuevos tweets al conjunto.

Este repositorio está en funcionamiento desde el 3 de abril de 2023 y en el podemos encontrar un total de veinte columnas donde se almacena información como: fecha de creación, numero de likes, numero de retweets, texto o idioma. Además también se encuentra información del usuario el cual ha realizado la publicación pudiendo ver datos como: fecha de creación de la cuenta, numero de seguidores, descripción... Algunos de los datos a destacar de estas columnas se muestran a continuación en las siguientes imágenes:

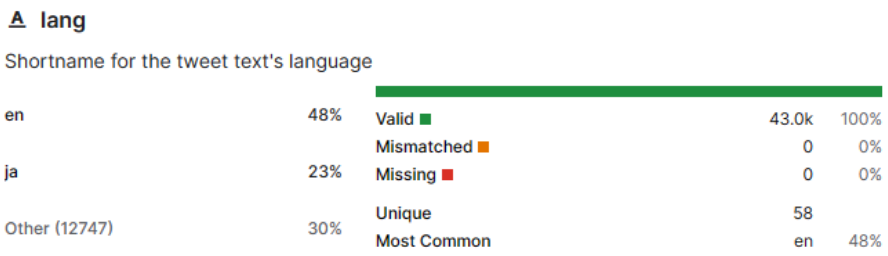


Figura 3.1: Información referida a idioma presentes

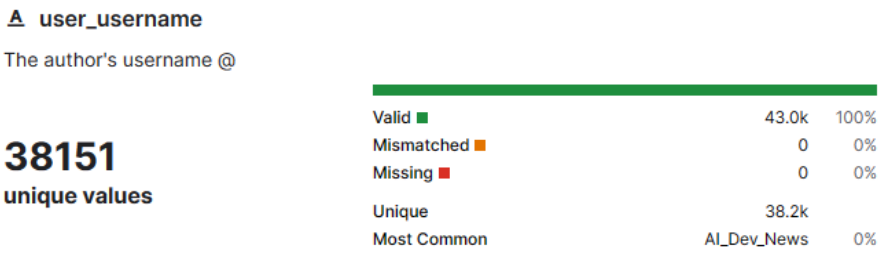


Figura 3.2: Información referida a nombres de usuario

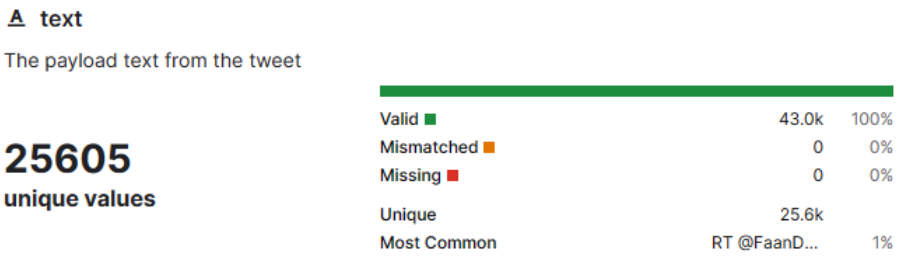


Figura 3.3: Información referida al texto del tweet

3.2. Python

Python es un lenguaje de programación de alto nivel y con capacidad de ser orientado a objetos. Su principal característica es su sintaxis clara y legible, lo que facilita la escritura y comprensión del código.

Además, Python cuenta con otras caracteristicas como:

- **Interpretado:** Este utiliza un enfoque de ejecución en tiempo de ejecución, lo que significa que el código se interpreta y ejecuta línea por línea en lugar de compilarlo en un archivo ejecutable. Esto permite una mayor flexibilidad y facilidad para probar y depurar el código.
- **Multiplataforma:** Python es compatible con múltiples sistemas operativos, incluyendo Windows, macOS y Linux. Esto significa que un programa escrito en Python puede ejecutarse en diferentes plataformas sin necesidad de cambios significativos.
- **Amplia biblioteca estándar:** Python cuenta con una amplia biblioteca estándar que abarca diversas áreas, como manipulación de archivos, redes, matemáticas, procesamiento de texto, desarrollo web y mucho más. Esto facilita el desarrollo de aplicaciones sin necesidad de buscar bibliotecas externas para tareas comunes.
- **Comunidad activa:** Python cuenta con una gran comunidad de desarrolladores que contribuyen con nuevas bibliotecas, frameworks y recursos adicionales. Esto significa que hay una amplia gama de herramientas disponibles para casi cualquier tarea que desees realizar.

Figura 3.4: Icono de Python

Entre sus aplicaciones más comunes, Python se utiliza en una variedad de áreas, como desarrollo web, análisis de datos, aprendizaje automático (machine learning), inteligencia artificial, automatización de tareas, scripting y más.

3.3. Twitter API

Versión gratuita que solo te permite publicar y versión de pago que te permite realizar consultas.

La API de Twitter es un conjunto de herramientas que permite a los desarrolladores acceder y manipular los datos de la red social. La API se divide en diferentes endpoints, que son las direcciones url que se usan para solicitar o enviar información. Algunos de los endpoints más comunes son los que permiten obtener los tweets de un usuario, buscar tweets por palabras clave, publicar tweets, seguir o dejar de seguir a otros usuarios, etc.

Figura 3.5: Icono de Twitter API

Esta API cuenta con diferentes niveles de subscripción según el nivel de acceso y el número de solicitudes que se quieran hacer.

El **plan gratuito** ofrece un acceso básico a la API, con un límite de 1500 mensajes publicados por mes pero sin opción a poder realizar consultas de tweets.

El **plan basic** afrece un mayor acceso a las funcionalidades de twitter, respecto al número de solicitudes el límite asciende hasta 10000 solicitudes de tweets por mes y un precio de 100USD mensuales.

El plan **pro** ofrece una gran diferencia en cuanto a números con el anterior plan ya que ahora el limite de tweets consultados/descargados asciende a 1 millón por mes, el precio de este plan tambien asciende a un total de 5000USD/mes .

A continuación en la figura 3.6 se puede observar aquellas diferencias más importantes entre los diferentes planes.

	Free	Basic	Pro
Getting access	Get Started	Get Started	Get Started
Price	Free	$100/month	$5000/month
Access to Twitter API v2	✔ (Only Tweet creation)	✔	✔
Access to standard v1.1	✔ (Only Media Upload and Login With Twitter)	✔ (Only Media Upload and Login With Twitter)	✔ (Only Media, Help, Rate Limit, and Login with Twitter)
Project limits	1 Project	1 Project	1 Project
App limits	1 App per Project	2 Apps per Project	3 Apps per Project
Tweet caps - Post	1,500	3,000	300,000
Tweet caps - Pull	✘	10,000	1,000,000
Filteres stream API	✘	✘	✔
Access to full-archive search	✘	✘	✔
Access to Ads API	✔	✔	✔

Figura 3.6: Planes de subscripción de la API

3.4. Emotion English DistilRoBERTa-base

Este modelo, el cual forma parte del repositorio de *Hugging Face* 3.7, está especializado en clasificar emociones en textos escritos en inglés. La clasificación se basa en diferenciar en los textos las 6 emociones básicas de Ekman (ira, asco, miedo, alegría, tristeza y sorpresa) y una neutral [35].

Figura 3.7: Icono de Huggin Face

Este es un modelo optimizado de RoBERTa (*Robusted Optimized Bidirectional Encoder Representations from Transformers*) el cual utiliza la red neuronal *Transformer* comentada anteriormente en el apartado 2.2.3.

El entrenamiento se ha realizado con el 80 % de un dataset balanceado de aproximadamente 2811 textos por emoción, haciendo un total de aproximadamente 20000 textos utilizados. Para ello se ha utilizado un subconjunto de 6 *datasets* diferentes los cuales cuentan con el texto etiquetado con la emoción a la que pertenecen. El 20 % de datos restantes se han utilizado para la evaluación del modelo en el que la precisión de este se ha estimado en el 66 %

En la siguiente tabla se puede observar el nombre de los *datasets* así como que emociones especifican:

Name	anger	disgust	fear	joy	neutral	sadness	surprise
Crowdflower (2016)	Yes	-	-	Yes	Yes	Yes	Yes
Emotion Dataset, Elvis et al. (2018)	Yes	-	Yes	Yes	-	Yes	Yes
GoEmotions, Demszky et al. (2020)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ISEAR, Vikash (2018)	Yes	Yes	Yes	Yes	-	Yes	-
MELD, Poria et al. (2019)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SemEval-2018, EI-reg, Mohammad et al. (2018)	Yes	-	Yes	Yes	-	Yes	-

Figura 3.8: Datasets utilizados en el entrenamiento del modelo

3.5. Pandas

La librería Pandas es una biblioteca de Python utilizada principalmente en la ciencia de datos. Este proporciona estructuras de datos y herramientas de manipulación de datos de alto rendimiento.

Figura 3.9: Icono de Pandas

Respeto a las estructuras de datos, Pandas implementa dos nuevos conceptos:

- **Series:** es una estructura de datos unidimensional que puede contener cualquier tipo de datos, similar a una columna en una hoja de cálculo. Cada elemento en una Serie tiene una etiqueta llamada índice, que permite acceder y manipular los datos de manera eficiente.
- **DataFrames:** es una estructura de datos bidimensional, similar a una tabla en una base de datos o una hoja de cálculo. Consiste en una colección de Series que comparten el mismo índice, lo que facilita la manipulación y el análisis de datos tabulares.

Respecto a las funcionalidades, algunas de estas son:

- Realizar operaciones de limpieza, transformación, filtrado, agregación y análisis de datos.

- Permitir cargar datos desde varios formatos, como archivos CSV, Excel, SQL, etc...
- Ofrecer capacidades flexibles para indexación y selección de datos, manejo de valores nulos, fusión y concatenación de conjuntos de datos.

3.6. Seaborn

Seaborn es una biblioteca de visualización de datos en Python que se basa en la popular biblioteca Matplotlib. Está diseñada para generar visualizaciones atractivas y de alta calidad con menos código que Matplotlib.

Seaborn proporciona una interfaz de alto nivel para crear gráficos estadísticos, lo que la hace especialmente útil para el análisis exploratorio de datos y la generación de visualizaciones informativas y atractivas. Ofrece una variedad de estilos predefinidos y paletas de colores que pueden aplicarse fácilmente a los gráficos, lo que permite personalizar rápidamente la apariencia de las visualizaciones.

Además de sus estilos y paletas, Seaborn proporciona funciones de trazado optimizadas para visualizar relaciones estadísticas, como diagramas de dispersión, gráficos de distribución, gráficos de barras y diagramas de cajas. Estas funciones permiten explorar patrones y relaciones en los datos de forma rápida y sencilla.

Seaborn también tiene integración con Pandas, otra popular biblioteca de análisis de datos en Python. Esto significa que puedes trabajar fácilmente con estructuras de datos de Pandas al crear visualizaciones con Seaborn.

En resumen, Seaborn es una biblioteca poderosa y fácil de usar que complementa a Matplotlib al proporcionar estilos atractivos y funciones estadísticas optimizadas para visualizaciones de datos en Python.

Figura 3.10: Icono de Seaborn

3.7. Matplotlib

Matplotlib es una popular biblioteca de visualización de datos en Python que te permite crear una amplia gama de gráficos y diagramas estáticos, animados e interactivos. Proporciona un conjunto completo de herramientas para generar diversos tipos de gráficos, como gráficos de líneas, gráficos de dispersión, gráficos de barras, histogramas, gráficos circulares (tartas), entre otros.

Matplotlib es altamente personalizable, lo que te brinda control sobre la apariencia y el estilo de tus gráficos. Proporciona una amplia variedad de opciones para modificar colores, marcadores, estilos

de línea, etiquetas, ejes, cuadrículas y leyendas, entre otros elementos. Esta flexibilidad te permite crear visualizaciones atractivas e informativas para comunicar eficazmente tus datos.

La biblioteca está diseñada para funcionar sin problemas con otras bibliotecas de Python, como NumPy y Pandas, lo que facilita el análisis y la manipulación de datos. Matplotlib se integra bien con Jupyter Notebook y otros entornos de computación interactiva, lo que permite la exploración interactiva y visualizaciones dinámicas.

Con Matplotlib, puedes guardar tus gráficos en varios formatos, como PNG, PDF, SVG o formatos interactivos como HTML o GIFs animados. La biblioteca admite capacidades de trazado tanto en 2D como en 3D limitadas, lo que te permite representar datos en múltiples dimensiones.

Figura 3.11: Icono de Matplotlib

3.8. Regular Expressions (Re)

Esta biblioteca de Python es un módulo estándar que proporciona herramientas para el procesamiento de texto mediante expresiones regulares. ‘re‘ es la abreviatura de regular expression”(expresión regular en inglés), y permite buscar, extraer y manipular patrones específicos en cadenas de texto.

Las expresiones regulares son secuencias de caracteres que definen un patrón de búsqueda. Con ‘re‘, puedes utilizar estas expresiones regulares para realizar diversas operaciones, como buscar coincidencias, reemplazar texto, dividir cadenas y extraer partes específicas de una cadena de texto.

El módulo ‘re‘ proporciona una variedad de funciones y métodos útiles para trabajar con expresiones regulares. Algunas de las funciones y métodos más comunes incluyen:

- **‘re.search()‘:** busca un patrón en una cadena y devuelve el primer resultado encontrado.
- **‘re.match()‘:** busca un patrón al comienzo de una cadena y devuelve el primer resultado encontrado.
- **‘re.findall()‘:** encuentra todas las ocurrencias de un patrón en una cadena y devuelve una lista con los resultados.
- **‘re.sub()‘:** reemplaza todas las ocurrencias de un patrón en una cadena con un texto especificado.
- **‘re.split()‘:** divide una cadena en partes utilizando un patrón como separador.

Además, el módulo ‘re‘ proporciona una serie de caracteres especiales y secuencias de escape que te permiten definir patrones más complejos. Estos patrones pueden incluir coincidencias literales, clases de caracteres, repeticiones, alternancias y más.

En resumen, la biblioteca ‘re‘ de Python es una herramienta poderosa para el procesamiento de texto utilizando expresiones regulares. Te permite realizar operaciones avanzadas de búsqueda y manipulación de patrones en cadenas de texto, lo que resulta útil en diversas aplicaciones como análisis de texto, extracción de información y limpieza de datos.

Capítulo 4

Desarrollo del proyecto

A la hora de poder explicar con mayor facilidades las fases del algoritmo, se ha querido diferenciar en las siguientes etapas, las cuales serán explicadas más detenidamente en los siguientes subapartados:

- Inicialización y **captación** de datos
- **Análisis** de sentimientos
- Guardado y **correlación** de los resultados.

Para ayudar en la explicación de este algoritmo se cuenta con el diagrama de procesos, el cual se puede visualizar en la figura 5.2. Este muestra de forma simplificada los bloqueas previamente mencionados así como las principales subtareas realizadas en cada bloque.

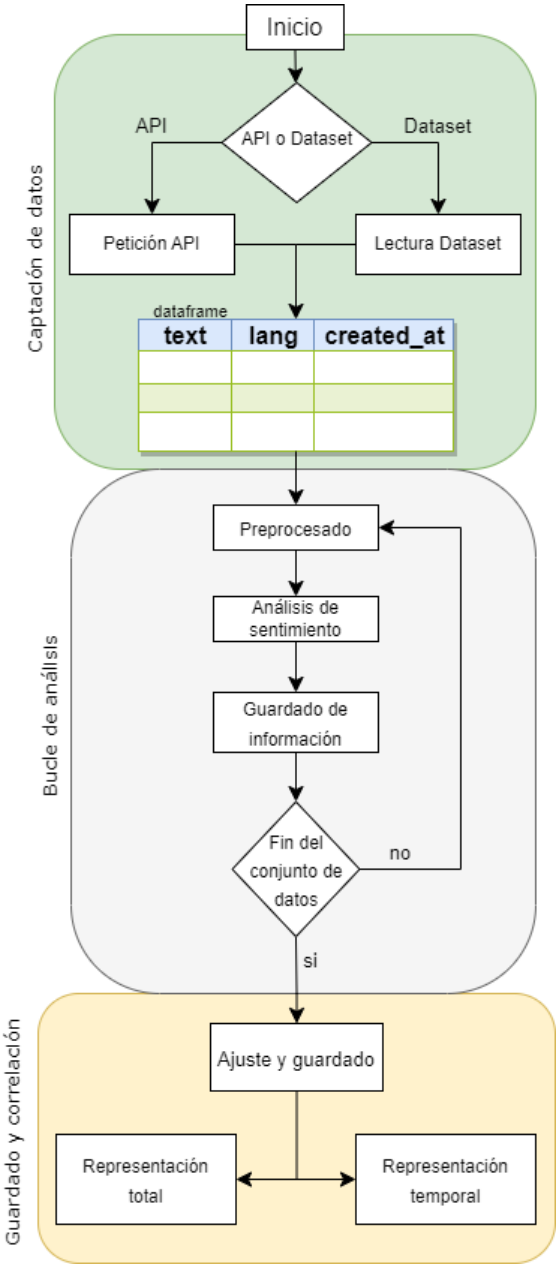


Figura 4.1: Diagrama de flujo del algoritmo desarrollado

4.1. Inicialización y captación de datos

Este apartado se centra en el primer bloque. Este consiste en obtener los tweets necesarios para el análisis de dos formas distintas para que posteriormente pueda ser posible el análisis de su información.

La **primera** de ellas se realiza mediante el uso de la API oficial de *twitter*. Para ello se necesita obtener un código identificador, llamado *Bearer Token*, de la pagina de desarrolladores de *twitter*.

Los pasos a seguir para obtener dicho código son:

- **Acceder a la portal:** Para ello habrá que iniciar sesión en *https://developer.twitter.com/en/portal/dashboard* con una cuenta de Twitter. Una vez realizado el login se podrá visualizar el portal mostrado en la figura 4.2

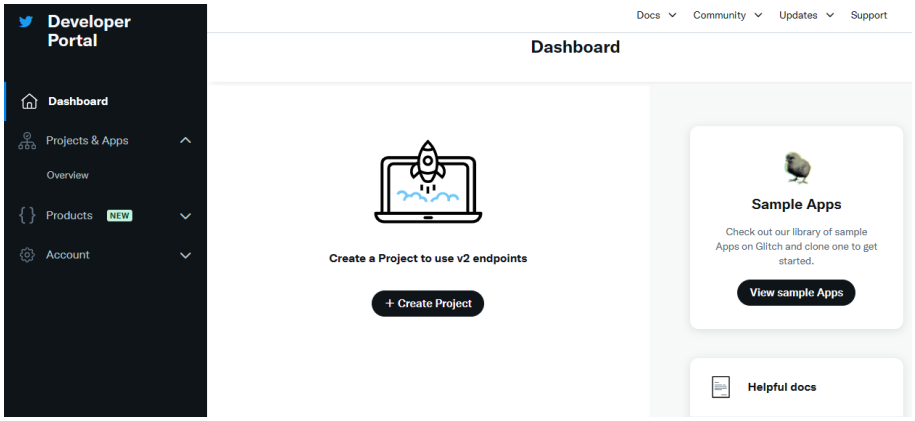


Figura 4.2: Portal de desarrolladores de Twitter

- **Dar de alta un proyecto y guardar su token** A continuación se realizará un proceso en el que se creará un proyecto nuevo y una aplicación ligada a ese proyecto. Para ello se deberá acceder a la opción de *Create Project* ubicada al centro de la figura 4.2

  Seguidamente se deberá rellenar el formulario especificando el nombre del proyecto, descripción de este y nombre de la aplicación, entre otros parámetros, como se muestra en la figura 4.3. Al final de este proceso se nos proporcionarán una serie de códigos y tokens entre los que se encontrará el *Bearer Token*. Una vez ubicado se deberá guardar en un sitio seguro para su posterior utilización. En caso de pérdida del token, la plataforma te permite regenerar un código diferente para la aplicación quedando inutilizado el token anterior.

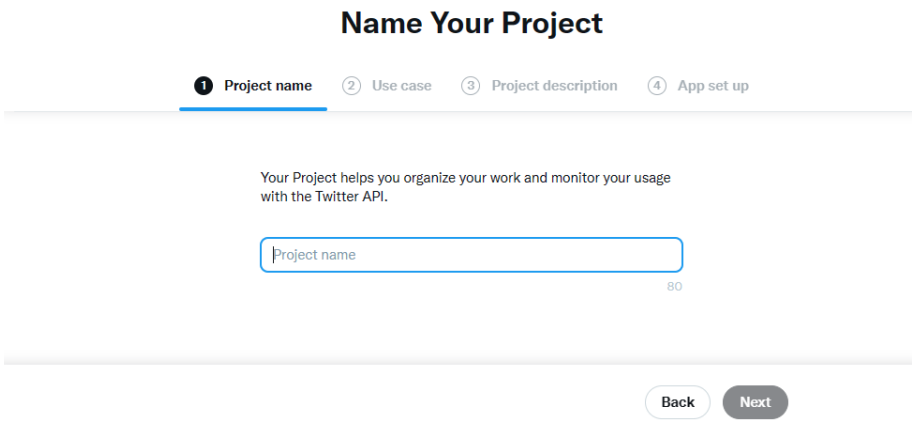


Figura 4.3: Proceso de creación de un nuevo proyecto/aplicación

Respecto a la **segunda** fuente de captación de tweets se realiza utilizando el *dataset* mencionado en el apartado 3.1. Para ello se debe descargar el conjunto de datos y tener en cuenta la ruta donde se guarda para especificarla próximamente en el código.

Una vez realizados estos pasos previos en los que se ha obtenido y token y se ha guardado el dataset en un directorio determinado, se procede a implementar el código necesario para el funcionamiento de este bloque.

En el extracto de código ubicado a continuación, se ha realizado el proceso de importar aquellas librerías que van a ser necesarias durante el proyecto y que han sido explicadas de forma teórica en el capítulo anterior. Adicionalmente se especifica en la variable *classifier* el modelo de inteligencia artificial que se va a utilizar para realizar la clasificación en emociones.

```
from transformers import pipeline #Para el analisis de emociones (IA)
import pandas as pd #Para los dataframes
import re   #Para el preprocesado de texto
import matplotlib.pyplot as plt #Para realizar plots
import seaborn as sns #Para realizar plots
import requests #Para la API de twitter

classifier = pipeline("text-classification",
model="j-hartmann/emotion-english-distilroberta-base",
top_k=None)
```

Listing 4.1: Importación de las librerias

Seguidamente se inicializan una serie de variables necesarias para el correcto funcionamiento del algoritmo, es en este momento donde se deberá pegar el token previamente guardado en la variable *bearer_token* así como especificar la ruta en la que se encuentra guardado el *dataset* que se va a utilizar. Adicionalmente contamos con otras variables como:

- *limit:* en la cual se especificará el numero de tweets que se desean analizar.
- *query:* en la cual se especifica que tema/hashtag se desea buscar en twitter (en el caso de utilizar esta opción)
- *search_url*: Dirección por el cual se realizaran las peticiones a Twitter, esta url viene proporcionada por la propia documentación de la API teniendo en cuenta el caso de uso.

```
#Ruta del dataset:
rutaDataset = r"C:\XXX\XXX\XXXX\XXX.csv"

#Numero de Tweets
limit = 43000

#Para peticiones en la API
query = "chatgpt"
bearer_token = "XXXXXXX"
search_url = "https://api.twitter.com/2/tweets/search/recent"
```

Listing 4.2: Inicialización de variables

En este extracto de código mostrado en el listing 4.3, se busca consultar al usuario que fuente quiere utilizar a la hora de captar los tweets.

Si el usuario **marca 1** el origen seleccionado será la de la API de twitter por lo que se le preguntará al usuario por la búsqueda del tema/hashtag que quiere realizar en la plataforma. Seguidamente ejecutará la función *tweetsAPI()* la cual devolverá un *dataframe* con toda la información obtenida de la petición que quedará guardada en la variable *data*.

Si por el contrario el usuario **marca 2**, se seleccionará para el análisis el *dataset* comentado anteriormente el cual contiene información limitada a un tema en concreto, por lo que se ejecutará la función *tweetsDataset()*

```
elegir = int(input(" 1- API Twitter \n 2- Dataset \n"))
match elegir:

    case 2:#DATASET
        data=tweetsDataset()

    case 1:#API
        query = input("Busqueda a realizar: ")
        data=tweetsAPI(query)
```

Listing 4.3: Algoritmo de elección de fuente de datos

La **función *tweetsAPI()*** es una función proporcionada por los propios desarrolladores de Twitter. El funcionamiento de esta función radica en realizar una consulta (*request*) al servidor teniendo un cuenta la dirección url del servidor, la autenticación y la busqueda del tema/hashtag que se quiere realizar (query). A continuación si el código de respuesta es favorable (200) se trasladará la información solicitada mediante la query a un dataframe

```
def tweetsAPI(query):

    query_params = {'query': query }
    response = requests.get(search_url, auth=bearer_oauth, params=query_params)
    print(response.status_code)
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    json_response = response.json()
    data = pd.json_normalize(json_response, record_path =['data'])
    return data

def bearer_oauth(r):

    r.headers["Authorization"] = f"Bearer {bearer_token}"
    r.headers["User-Agent"] = "v2RecentSearchPython"
    return r
```

Listing 4.4: Función encargada de realizar consultas a Twitter

Por otro lado, la **función *tweetsDataset()*** realizará una carga de los datos del *dataset* a un *dataframe* teniendo en cuenta la ruta donde se encuentra el archivo así como la cantidad de lineas (*tweets*) que se quieren leer.

```
def tweetsDataset():

        data = pd.read_csv(rutaDataset,nrows=limit)
        print(data.columns)
        return data
```

Listing 4.5: Función encargada de la lectura de archivos .csv (datasets)

En este punto de la ejecución del programa e independientemente de la fuente de información elegida, contaremos con un dataframe almacenado en la variable *data* con varias columnas entre las que se destacan:

- *lang*: **Idioma** en el que se ha escrito el tweet.
- *text*: Texto del **tweet**.
- *created_at*: **Fecha** y hora a la que el tweet ha sido publicado.

4.2. Análisis de sentimientos

Primeramente se descartarán entradas que pueden estar duplicadas en el dataframe. Además, debido a las limitaciones de idioma del modelo de clasificación elegido, se descarta todos aquellos textos que no estén escritos en inglés prestando atención a la columna *lang*.

Seguidamente se procederá a analizar el dataframe de forma iterativa. En cada iteración se ejecutarán funciones para el preprocesado del texto, análisis de sentimientos, así como el almacenamiento del análisis en variables auxiliares de cada una de las entradas de dicho conjunto de datos. Al terminar todas las iteraciones, la información será guardada en un dataframe que contará con las columnas: *texto*, *sentimiento*, *scores* (probabilidad de acierto) y *fecha*

```
#Eliminamos entradas duplicadas
data = data.drop_duplicates()


for indice, entrada in data.iterrows():

    if entrada["lang"]=="en":

        pre = preprocesado(entrada["text"]) #Preprocesado del texto
        emot,score = emotion(pre)   #Analisis del sentimiento devolvieno
    sentimiento y probabilidad
        emotions.append(emot)   #Añadir sentimiento a la lista
        scores.append(score)    #Añadir porcentaje a la lista

        #Guardamos toda la información en la lista
        tweets.append([pre,emot,score,entrada["created_at"]])

#Creamos el data frame con la lista y añadimos los nombres de las cabeceras
df = pd.DataFrame(tweets, columns=['texto',"sentimiento","scores","fecha"])
```

Listing 4.6: Bucle de analisis del conjunto de datos

Respecto a la función de preprocesado, esta recibe el tweet mediante la variable *frase*. A continuación y utilizando la librería Re, descrita en el cápitulo anterior, en el apartado 3.8, se procede a sustituir las menciones por "*@user*" y los links por "*http*". Además, se eliminan los posibles saltos de linea de forma que todo el texto no esté en varios párrafos diferentes y los hashtags. De esta forma la función se deshace de toda esa información personal o de poca utilidad para el análisis.

```
def preprocesado(frase):
```

```
    # Eliminar menciones
    frase = re.sub(r'@[^\s]+', '@user', frase)
    # Eliminar hastags
    frase = re.sub(r'\B#\S+', '', frase)
    # Eliminar URLs
    frase = re.sub(r'http\S+', 'http', frase)
    # Eliminar saltos de linea
    frase = re.sub(r'\n', ' ', frase)
    return frase.lower()
```

Listing 4.7: Preprocesado del texto

A continuación la frase preprocesada por la función previamente descrita será enviada al modelo para su análisis y clasificación de emociones. Para ello se utiliza el modelo previamente inicializado. Esta devolverá dos datos importantes para el análisis los cuales son el nombre de la emoción y la probabilidad de acierto.

```
def emotion(frase):
    a = classifier(frase)
    emotion = a[0][0]
    print(emotion)
    return emotion.get("label"),emotion.get("score")
```

Listing 4.8: Clasificación en emociones

4.3. Guardado y correlación

Como se ha comentado en el apartado anterior, al iniciar este bloque del algoritmo, contaremos con un dataframe con información del tweet (texto y fecha) así como del análisis (emoción y probabilidad de acierto). En este bloque primeramente se realizan un par de tareas relacionadas con la fecha es decir, se le da el formato de fecha a la columna que guarda dicha información así como ordenar el dataframe por esta misma variable.

A continuación se guarda el resultado de todo el análisis, contenido en el dataframe, en un fichero .csv para que el usuario pueda visualizar los resultados en herramientas especializadas como puede ser el caso de *Microsoft Excel*.

Una vez realizado esto, se procede a ejecutar las funciones plot_sentiment() y plot_lineal(), las cuales realizaran dos representaciones gráficas diferentes de la información procesada.

Respecto la primera función, se le facilitan una serie de parámetros como el título, subtitulo, descripción de la gráfica e incluso media de las probabilidades del análisis.

```
#Ajustamos fecha
df['fecha'] = pd.to_datetime(df['fecha'])
df = df.sort_values('fecha')

#Guardamos el resultado en .csv
df.to_csv(r'resultado.csv', index=False)

#Creamos etiquetas con los nombres de cada emoción y realizamos los plots
labels = ["fear","neutral","joy","sadness","anger","disgust","surprise"]
plot_sentiment(df,labels, title='Resultado total del analisis',
```

```
                subtitle='Emociones ordenadas de mayor a menor valor',
                description="Ratio de acierto del analisis: "+ str(df['scores'].
    mean()))
plot_linear(df,labels)
```

Listing 4.9: Guardado de información del analisis y llamada a las función de representación gráfica

Respecto a plot_sentiment() es una función que realizará un **gráfico de barras** donde cada emoción será una barra diferente, mostrando así el numero total de tweets clasificados en cada emoción. Una de las subtareas a destacar de esta función es el conteo que se realiza al principio de este con el que se pretende obtener el numero de mensajes por emoción que hay. Adicionalmente, al final contamos con un comando que guardará la gráfica generada en el directorio de trabajo bajo el nombre de *Barras.png*

```
def plot_sentiment(df, labels, title='', subtitle='', description='', palette='
    bright'):
    #Contar el número de sentimientos que hay
    conteo = df["sentimiento"].value_counts()
    neutral = conteo.get("neutral",0)
    joy = conteo.get("joy", 0)
    surprise = conteo.get("surprise", 0)
    fear = conteo.get("fear", 0)
    sadness = conteo.get("sadness", 0)
    anger = conteo.get("anger", 0)
    disgust = conteo.get("disgust", 0)

    y = [fear, neutral, joy, sadness, anger, disgust, surprise]

    # Ordenar los datos por los valores de 'y' en orden descendente
    data = sorted(zip(labels, y), key=lambda x: x[1], reverse=True)
    x_sorted = [item[0] for item in data]
    y_sorted = [item[1] for item in data]

    sns.set(style='whitegrid')

    # Crear un gráfico de barras horizontales utilizando seaborn
    g = sns.barplot(x=y_sorted, y=x_sorted, palette=palette, orient='h',
    linewidth=0.5)

    g.set_xlabel('')
    g.set_ylabel('')

    # Configurar el título del gráfico
    g.set_title(
        f'{title}'.upper(),
        loc='left',
        fontdict=dict(
            fontsize=15,
            fontweight='bold'),
        pad=30)

    # Configurar las etiquetas del eje x
    g.set_xticklabels(
        [f' {tick_label.get_text().title()}' for tick_label in g.
    get_xticklabels()],
        fontdict=dict(
```

```
            fontsize=12.5,
            fontweight='medium'))

    # Agregar el subtítulo en la parte superior del gráfico
    plt.text(s=f'{subtitle}',
             alpha=0.5,
             x=0,
             y=1.04,
             horizontalalignment='left',
             transform=g.transAxes)

    # Agregar la descripción en la parte inferior del gráfico
    plt.text(s=f'{description}',
             alpha=0.5,
             x=0,
             y=-0.15,
             verticalalignment='top',
             horizontalalignment='left',
             transform=g.transAxes)

    # Agregar etiquetas con los valores de cada barra
    for i, v in enumerate(y_sorted):
        g.text(v, i, str(v), ha='left', va='center', fontdict=dict(color='black
    ', fontsize=12, fontweight='medium'))

    # Ajustar el diseño del gráfico para que se muestre correctamente
    plt.tight_layout()

    # Guardar el gráfico como una imagen PNG
    plt.savefig('Barras.png')

    # Mostrar el gráfico en la pantalla
    plt.show()
```

Listing 4.10: Función de representación total de emociones

La última función a analizar se trata de plot_linear() esta función es la encargada de representar un subplot por cada emoción correlacionandolo con las fechas de publicación. De esta forma en el eje X contaremos con las fechas del conjuntos de tweets analizados pudiendo así visualizar de forma gráfica cuantos tweets ha habido por día o que tendencias han surgido en cada sentimiento a lo largo de los días.

La primera tarea de esta función trata de agrupas las horas en el intervalo escogido, de esta forma se puede buscar tener más o menos precisión en la visualización. A continuación se encontrarán los comandos necesarios para la creación de dicha gráfica.

```
def plot_linear(df,labels):

    df['fecha'] = df['fecha'].dt.floor('24H')
    # Pivotar el DataFrame para asegurar que haya una columna para cada
    sentimiento en cada fecha
    df_pivot = df.pivot_table(index='fecha', columns='sentimiento', aggfunc='
    size', fill_value=0)
    df_pivot = df_pivot.reindex(columns=labels, fill_value=0)

    # Contar el número de sentimientos por fecha y categoría
    df_counts = df_pivot.stack().reset_index(name='count')
    print("Fecha ordenada: \n")
```

```
    print(df["fecha"])

    # Obtener el rango de fechas del DataFrame
    fecha_min = df_counts['fecha'].min()
    fecha_max = df_counts['fecha'].max()

    # Crear una figura con subplots para cada sentimiento
    fig, axes = plt.subplots(len(labels), 1, figsize=(10, 12), sharex=True)

    # Definir una paleta de colores para los sentimientos
    colores = sns.color_palette('husl', n_colors=len(labels))

    # Iterar sobre los sentimientos y crear una gráfica de líneas en cada
    subplot
    for i, sentimiento in enumerate(labels):
        df_sentimiento = df_counts[df_counts['sentimiento'] == sentimiento]
        df_sentimiento = df_sentimiento.set_index('fecha')


        sns.lineplot(data=df_sentimiento, x=df_sentimiento.index, y='count', ax
    =axes[i], color=colores[i])
        axes[i].set_title('')  # Eliminar el título arriba de cada gráfica
        axes[i].tick_params(axis='x', labelrotation=0)
        axes[i].set_ylim(0, df_counts['count'].max())  # Establecer los límites
     del eje y

        # Agregar grid de líneas horizontales
        axes[i].grid(axis='y', linestyle='--', linewidth=0.5)

        # Establecer los límites del eje x para mostrar solo las fechas del
    DataFrame
        axes[i].set_xlim(fecha_min, fecha_max)

        # Etiqueta del eje y con el nombre del sentimiento y la escala
        axes[i].set_ylabel(f'{sentimiento}')

    # Configurar el título y etiquetas del eje x del último subplot
    axes[-1].set_xlabel('Fecha')

    # Ajustar los espacios entre los subplots
    plt.tight_layout()

    # Ajustar el parámetro bottom de la figura
    plt.subplots_adjust(bottom=0.1)

    #Guardamos figura
    plt.savefig('Lineas.png')
    # Mostrar la figura
    plt.show()
```

Listing 4.11: inicialización de variables

Capítulo 5

Resultados

A la hora de ejecutar el algoritmo, se ha decidido utilizar el dataset como fuente para la obtención de tweets y se ha fijado el numero de tweets a analizar a 43000 lo cual se espera ser una cifra suficiente para que pueda contener información interesante.

Durante el proceso, se ha ido pudiendo observar el análisis en tiempo real ya que aparecían por terminal los resultados de la clasificación de cada uno de los tweets analizados como se muestra en el extracto 5.1

```
{'label': 'neutral', 'score': 0.8298044800758362}
{'label': 'joy', 'score': 0.3141423165798187}
{'label': 'joy', 'score': 0.6520513892173767}
{'label': 'joy', 'score': 0.39887186884880066}
{'label': 'fear', 'score': 0.9466402530670166}
{'label': 'joy', 'score': 0.7760977745056152}
{'label': 'anger', 'score': 0.3666483759880066}
{'label': 'sadness', 'score': 0.4187062978744507}
{'label': 'fear', 'score': 0.6352677941322327}
{'label': 'fear', 'score': 0.3987298309803009}
```

Listing 5.1: inicialización de variables

Una vez terminado el bloque de análisis y como se comentó en el capítulo de desarrollo, el algoritmo guardará los resultados en un fichero con formato .csv.

Para visualizarlo de una forma más cómoda a como se haría abriendo el fichero con un editor de texto plano, se procede ejecutar Microsoft Excel, crear una hoja en blanco y importar el archivo .csv. Una vez realizado este rápido proceso se podrá visualizar una tabla la cual cuenta con funcionalidades de interés como es el filtrar. Con esto se pueden filtrar por solo aquellas entradas con un determinada emoción, fecha...

	A	B	C	D
1	texto	sentimiento	scores	fecha
2	hey guys, has launched at just $ 55,000 usd marketcap. the chatgpt of dex - reimagining defi w	surprise	0,431747556	03/04/2023 2:59
3	@user chatgpt	neutral	0,832023919	03/04/2023 2:59
4	rt @user we are listing on our own exchange and sushiswap at 2:30 pm utc zenithswap - the cha	neutral	0,748865128	03/04/2023 2:59
5	i wish was only about creatives losing their jobs. goldman sachs says 300 million jobs will be aff	anger	0,458843082	03/04/2023 2:59
6	rt @user http order your custom that look like designed to look like a	neutral	0,382269204	03/04/2023 2:59
7	rt @user massive update for auto-gpt: code execution! auto-gpt is now able to write it's	neutral	0,401436269	03/04/2023 2:59
8	rt @user italy bans @user joining russia, china, north korea, and iran. learn more: http | @use	neutral	0,364802331	03/04/2023 2:59
9	rt @user i asked the new chatgpt browsing extension to find me some money. within a minute,	surprise	0,778948009	03/04/2023 2:59
10	this article examines the ability of gpt-4, a powerful new ai language model, to accurately asses	neutral	0,599121213	03/04/2023 2:59
11	i'm going to completely use chatgpt for this.. hahah!! http	joy	0,754765034	03/04/2023 2:59
12	(2/6) here's a recent standout prompt from the library forum it guides users to create well-struc	joy	0,776097775	03/04/2023 2:59

Figura 5.1: Tabla de los datos exportados

Seguidamente se genera la gráfica de barras encargada de mostrar la suma de tweets por sentimiento. En esta se puede observar también la media de las probabilidades de todo el conjunto, es decir la probabilidad de acierto total del análisis. En este caso, la probabilidad resultante ha sido de un 60,66 % lo cual se asemeja a la probabilidad que los creadores del modelo de clasificación de emociones habían estimado, siendo este de un 66 %

Además, cabe destacar que la suma total del número de tweets clasificados va a ser inferior al analizado (43000 tweets). Esto se debe a los tweets duplicados y tweets en otros idiomas que contiene el dataset. Si para esta prueba se utilizase la API de twitter, se podría añadir un criterio de búsqueda con ciertos filtros en la variable query que limitase el idioma a inglés, de esta forma la diferencia se vería drásticamente reducida.

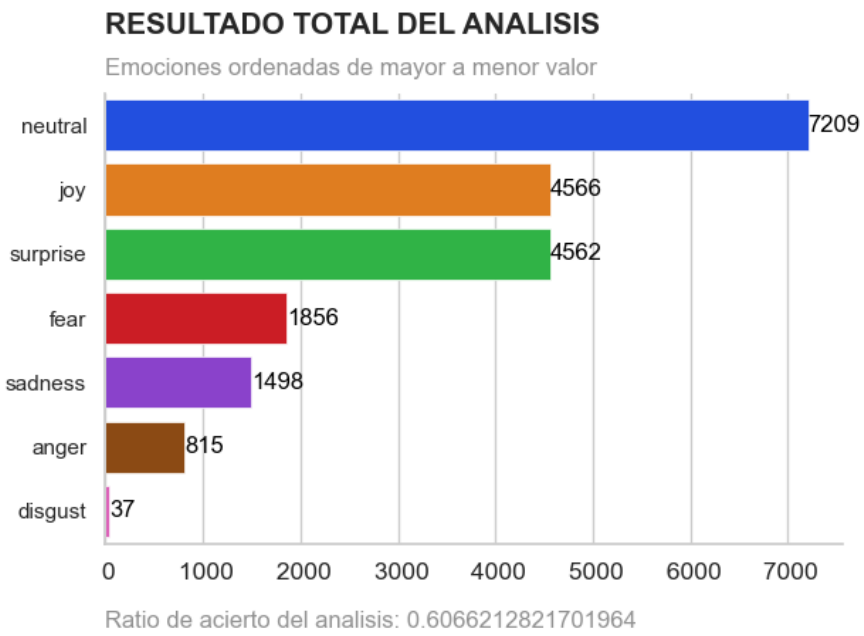


Figura 5.2: Representación clasificación en emociones

A continuación, en la figura 5.3 , se muestra la gráfica temporal en la que se ha decidido agrupar en los tweets en intervalos de 24 horas. En esta representación se puede observar que el día 6 de Abril hubo un máximo en cuanto a publicaciones relacionadas con Chatgpt.

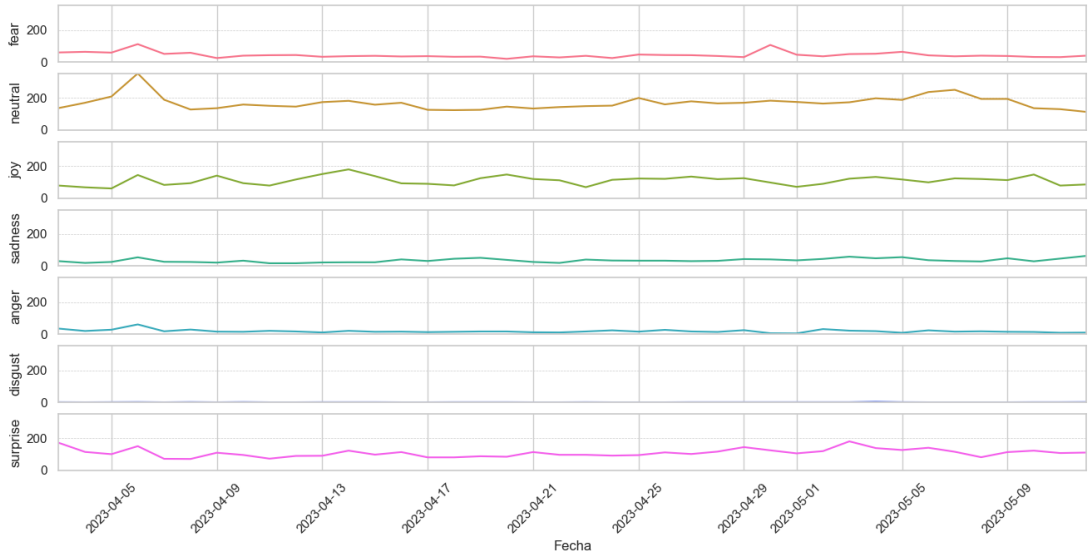


Figura 5.3: Representación temporal de las emociones

Como se había comentado anteriormente la resolución se podía variar, cambiando de valor de cierta función presente en el extracto de código 4.11. De esta forma el usuario tiene libertad de escoger aquella resolución que se adapta más a sus necesidades.

En la figura 5.4 podemos ver el como la resolución de esta ha variado a 1 hora pudiendo obtener un mayor grado de detalle.

Figura 5.4: Representación temporal de las emociones

También hay que tener en cuenta que la gráficas generadas permiten hacer zoom en aquellos sectores elegidos para su mejor visualización. Esta y otras opciones solo están disponibles cuando el compilador muestra la imagen pero no una vez cerrada la ventana. Estas opciones se pueden visualizar en la parte inferior de la figura 5.5.

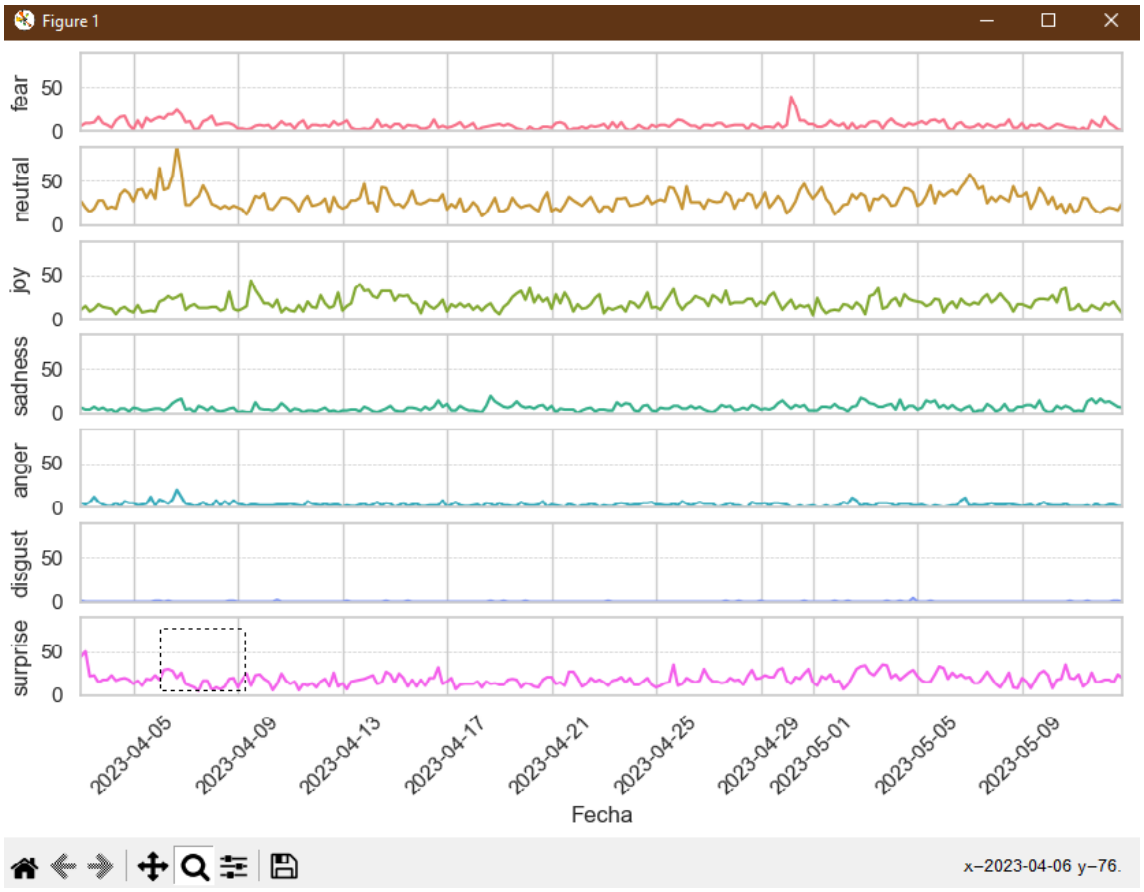


Figura 5.5: Visualizador de gráficas

Capítulo 6

Conclusiones y lineas futuras

Como conclusión, se puede destacar que se ha conseguido implementar un algoritmo de inteligencia artificial capaz de clasificar los tweets en 7 emociones diferentes.

Para ello se han programado diferentes funciones para que los tweets puedan ser obtenidos tanto de datasets como desde la API oficial de twitter realizando las peticiones pertinentes.

Además de clasificar las emociones, se ha añadido la funcionalidad de poder visualizar tanto el computo total de tweets clasificados en cada emoción así como una vista temporal pudiendo verse reflejadas posibles tendencias.

Por último, se ha realizado una prueba utilizando el dataset descrito en el apartado 3.1 en el cual se recogían más de 43000 tweets relacionados con ChatGPT. Se ha podido ver el proceso de este algoritmo mostrando diferentes resultados como la clasificación única por tweet, el archivo .csv generado, así como la gráfica donde se especificaba la suma de cada emoción y la probabilidad de acierto total o la gráfica temporal.

Como conclusiones de este análisis en particular, se podría comentar que la principal emoción envuelta, a la hora analizar las opiniones que los usuarios de twitter han tenido respecto a ChatGpt, ha sido la neutralidad. Esto implica textos como por ejemplo de carácter informativo y abordados de forma generalmente objetiva. Seguidamente y con valores bastante similares se encuentran las emociones de alegría y sorpresa, por lo que se podría determinar que, en el intervalo de tiempo contemplado por el dataset, el feedback obtenido por los usuarios ha sido positivo.

Respecto a lineas futuras, algunas de ellas serian las siguientes:

Posibilidad de añadir traducción

Como se ha comentado durante el desarrollo del proyecto, este modelo de inteligencia artificial solo trabaja con textos en inglés, dado que actualmente no hay ningún modelo que clasifique en 7 emociones y con cualquier idioma. Una opción seria implementar un algoritmo capaz de traducir de los principales idiomas (o de todos) al inglés para poder realizar la clasificación de estos textos y no perder información importante.

Detector de ironía y discursos de odio

Otras de las tareas del NLP es la detección de estos fenómenos, hay modelos que están especializados en estos casos y que añadiéndolos a este algoritmo se podría enriquecer la información proporcionada al usuario.

Bibliografía

[1] *Teoría de las EMOCIONES de PAUL EKMAN - Resumen y conclusiones*. https://www.psicologia-online.com/teoria-de-las-emociones-de-paul-ekman-5391.html. (Accessed on 07/02/2023).

[2] *Marketing emocional: claves, implementación y ejemplos de éxito*. https://resources.esmartia.com/blog/marketing-emocional-claves-implementacion-y-ejemplos-de-exito. (Accessed on 06/22/2023).

[3] A. M. TURING. "I.—COMPUTING MACHINERY AND INTELLIGENCE". En: *Mind* LIX.236 (oct. de 1950), págs. 433-460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

[4] *Aplicaciones reales de la inteligencia artificial en la medicina | APD*. https://www.apd.es/aplicaciones-inteligencia-artificial-en-medicina/. (Accessed on 06/08/2023).

[5] *Mamografía Bilateral 3D con Tomosíntesis en Madrid*. https://www.agrupacionginecologica.es/es/contrata-online/pruebas-mamarias-6/mamografia-bilateral-3d-con-tomosintesis-22. (Accessed on 06/08/2023).

[6] *Aplicaciones de inteligencia artificial (IA) en la educación*. https://www.educo.org/blog/aplicaciones-de-ia-en-la-educacion. (Accessed on 06/08/2023).

[7] *Las 4 funciones de la educación más importantes*. https://www.lifeder.com/funciones-educacion/. (Accessed on 06/08/2023).

[8] *7 Aplicaciones de la Inteligencia Artificial en la Ingeniería Industrial*. https://www.edsrobotics.com/blog/aplicaciones-inteligencia-artificial-en-ingenieria-industrial/. (Accessed on 06/12/2023).

[9] *Pick and Place Robots - How do Robotic Arms Help Industries in Pick and Place of Products? - iCharts*. https://www.icharts.net/pick-and-place-robots-how-do-robotic-arms-help-industries-in-pick-and-place-of-products/. (Accessed on 06/12/2023).

[10] *The three different types of Artificial Intelligence – ANI, AGI and ASI – EDI Weekly: Engineered Design Insider*. https://www.ediweekly.com/the-three-different-types-of-artificial-intelligence-ani-agi-and-asi/. (Accessed on 05/26/2023).

[11] *¿Qué es un árbol de decisión? | IBM*. https://www.ibm.com/es-es/topics/decision-trees. (Accessed on 06/13/2023).

[12] *All You Need to Know About Support Vector Machines*. https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/. (Accessed on 06/13/2023).

[13] *Los 10 modelos más populares de inteligencia artificial*. URL: https://www.clubdetecnologia.net/blog/2018/los-10-modelos-mas-populares-de-inteligencia-artificial/.

[14] *One Hot Encoding | Interactive Chaos*. https://interactivechaos.com/es/manual/tutorial-de-machine-learning/one-hot-encoding. (Accessed on 06/13/2023).

[15] *Term Frequency Inverse Document Frequency (TF-IDF) - Visitor Analytics*. https://www.visitor-analytics.io/es/glosario/t/term-frequency-inverse-document-frequency-tf-idf/. (Accessed on 06/13/2023).

[16] Nhan Cach Dang, María N. Moreno-García y Fernando De la Prieta. "Sentiment Analysis Based on Deep Learning: A Comparative Study". En: *Electronics* 9.3 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9030483. URL: https://www.mdpi.com/2079-9292/9/3/483.

[17] *Natural language processing - Documentación de IBM*. https://www.ibm.com/docs/es/rpa/21.0?topic=automation-natural-language-processing. (Accessed on 05/25/2023).

[18] Steven W. Smith. *Machine translation of languages: Fourteen essays*. MIT Press., 1955, pág. 243. ISBN: 9780262120029. URL: https://mitpress.mit.edu/9780262120029/machine-translation-of-languages/.

[19] Lawrence R. Rabiner y Biing-Hwang Juang. "Fundamentals of speech recognition". En: *Prentice Hall signal processing series*. 1993.

[20] *frontmatter.pdf*. https://sigir.org/files/museum/introduction_to_modern_information_retrieval/frontmatter.pdf. (Accessed on 05/26/2023).

[21] Inderjeet Mani y Mark T. Maybury. *Advances in Automatic Text Summarization*. https://mitpress.mit.edu/9780262133593/advances-in-automatic-text-summarization/. 1999.

[22] Ilya Sutskever, Oriol Vinyals y Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". En: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[23] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". En: 2019.

[24] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". En: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, nov. de 2016, págs. 2383-2392. DOI: 10.18653/v1/D16-1264. URL: https://aclanthology.org/D16-1264.

[25] *Recurrent Neural Network (RNN): ¿de qué se trata?* https://datascientest.com/es/recurrent-neural-network-rnn-de-que-se-trata. (Accessed on 06/14/2023).

[26] *An Intuitive Explanation of LSTM. Recurrent Neural Networks | by Ottavio Calzone | Medium*. https://medium.com/\spacefactor\@m{}ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. (Accessed on 06/14/2023).

[27] Ashish Vaswani et al. "Attention is all you need". En: *Advances in neural information processing systems* 30 (2017).

[28] Siyuan Du y Hao Wang. “Addressing Syntax-Based Semantic Complementation: Incorporating Entity and Soft Dependency Constraints into Metonymy Resolution”. En: *Future Internet* 14 (mar. de 2022), pág. 85. DOI: 10.3390/fi14030085.

[29] Red Hat. *What are application programing interfaces*. URL: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces#history-of-apis.

[30] *Qué es una API REST, para qué sirve y ejemplos*. https://blog.hubspot.es/website/que-es-api-rest. (Accessed on 05/11/2023).

[31] *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)*. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. (Accessed on 05/11/2023).

[32] *Introducción a las APIs web - Aprende desarrollo web | MDN*. https://developer.mozilla.org/es/docs/Learn/JavaScript/Client-side_web_APIs/Introduction. (Accessed on 05/25/2023).

[33] *What are Payment APIs? The Future of Payments | MuleSoft Blog*. https://blogs.mulesoft.com/digital-transformation/business/what-are-payment-apis/. (Accessed on 05/25/2023).

[34] Enric Domingo. *ChatGPT 1000 Daily Tweets*. 2023. DOI: 10.34740/KAGGLE/DSV/5685262. URL: https://www.kaggle.com/dsv/5685262.

[35] Jochen Hartmann. *Emotion English DistilRoBERTa-base*. https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/. 2022.

Parte II

Anexos

Apéndice A

Listados adicionales