UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Análisis de la relevancia social de los temas tratados en los informativos de televisión de RTVE

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Christian Gilabert Oltra

Tutor: Diego Álvarez Sánchez

Curso: 2017-2018

Als meus iaios.

Resumen

El objetivo de este TFG es desarrollar una aplicación que analice la relevancia social de los telediarios de Radio Televisión Española mediante el análisis de frecuencias de las palabras que se dicen en ellos contra los tópicos de los que más se habla en las redes sociales. Esta herramienta facilitará tanto la obtención de los datos como su limpieza y permitirá realizar una comparación objetiva de estos de forma que el usuario pueda sacar conclusiones de forma sencilla y rápida.

Palabras clave: Telediario, análisis, frecuencias, redes sociales, relevancia, Python.

Resum

El objectiu d'aquest TFG es el desenvolupament d'una aplicació que analitze la rellevància social dels telediaris de Radio Televisió Espanyola mitjançant el analisi de les frequències de les paraules dites en aquestos contra els tópics que mes es parlen a les xarxes socials. Aquesta ferramenta facilitará tant la obtención de les dades com la seua neteja y permetrá realitzar una comparativa objectiva d'aquestos de manera que l'usuari puga traure conclusions de forma ràpida i senzilla.

Paraules clau: Telediari, análisi, frequències, xarxes socials, rellevància, Python.

Abstract

The objective of this TFG is to develop an application that analyzes the social relevance of the news programs of Radio Televisión Española through the analysis of the frequencies of the words that are spoken in them against the topics that are most talked about in social networks. This tool will facilitate both the collection of the data and its cleaning and will allow an objective comparison of these so that the user can draw conclusions easily and quickly.

Keywords: Newscast, analysis, frequencys, social networks, relevance, Python.

Índice

Índice de tablas

Índice de imágenes

Definiciones, abreviaturas y acrónimos

RTVE: La Corporación de Radio y Televisión Española, es una sociedad mercantil estatal que organiza la gestión indirecta del servicio público de radio y televisión de España.

Tema del momento (en inglés, *Trending Topic*, TT): Es una de las palabras o frases más repetidas en un momento concreto en una red social.

Tuit (en inglés, *tweet)*: Mensaje digital que se envía a través de la red social Twitter® y que no puede rebasar un número limitado de caracteres.

Bolsa De Palabras: Método que se utiliza en el procesado del lenguaje para representar documentos ignorando el orden de las palabras. En este modelo, cada documento parece una bolsa que contiene algunas palabras. Por lo tanto, este método permite un modelado de las palabras basado en diccionarios, donde cada bolsa contiene un conjunto de palabras del diccionario.

1. Introducción

Recientemente, a la fecha de realización de este proyecto han surgido diversas noticias sobre la supuesta imparcialidad de RTVE con relación a que tipos de noticias se comunican, qué forma tienen de contarlas y cuáles directamente se omiten y no son comunicadas al público en general.

Movimientos como *#MujeresTVE* o *#AsíSeManipula* (1) formados por integrantes del organismo de RTVE han denunciado que estos sucesos ocurren diariamente y asociaciones como Reporteros Sin Fronteras han mostrado su apoyo hacia estos movimientos (2) y han manifestado su malestar con las practicas que se llevan a cabo en la televisión pública.

Este movimiento ha denunciado prácticas como ocultar parte de una noticia para modificar su mensaje, dedicar más tiempo a ciertos partidos políticos que a otros o no dar una información objetiva sobre sucesos graves como conflictos o guerras son algunas de las prácticas que se denuncian (3).

El descontento respecto a esto se ha podido notar en las redes sociales donde se han sacado a la luz una serie de titulares que han resultado de interés a sus usuarios y que sin embargo no se han mencionado en los noticiarios (4). Este hecho resulta de especial relevancia si tenemos en cuenta que se trata de una televisión pública que se paga con los impuestos de todos los ciudadanos de este país y que por tanto debería de ser imparcial políticamente y objetiva en su discurso.

La cantidad de gente que usa estos noticiarios como fuente de información principal y que no puede o no tiene tiempo de contrastar o corroborar la información que se le presenta sigue siendo muy extensa (5) y, por tanto, se debería asegurar que esta información es relevante, objetiva y veraz.

Además, cabe destacar el reciente cambio de gobierno y toda la sucesión de despidos ocurridos en el panorama de la comunicación pública (6) han suscitado aún más recelos en la población lo que podría solucionarse con herramientas que analizarán los contenidos que se emiten y pudieran determinar de forma objetiva si los contenidos resultan relevantes.

1.1 Motivación

La escasez de herramientas que permitan un análisis objetivo del contenido de los noticiarios de RTVE y la urgente necesidad de esta suponen la principal motivación para realizar este proyecto, así como el aprendizaje y uso de distintas interfaces de programación de terceros que pueden resultar útiles para ello.

Personalmente, creo que un ente público que debe dar información a un país entero no debería estar controlado por ningún tipo de partido político pues se propicia la parcialidad de la información retransmitida y se incurre en una manipulación del telespectador.

Este hecho junto con todo lo mencionado en el apartado anterior fueron los que me animaron a unirme al proyecto que tenía en marcha mi tutor y, en concreto, a la parte de análisis de relevancia social. Me parece que la mejor forma de detectar si una noticia

resulta relevante para el espectador es comprobando que los temas que le interesan se tratan en las redes sociales.

1.2 Objetivos

El presente TFG forma parte de la solución general de análisis de telediarios. En concreto, se aborda el desarrollo de la aplicación de análisis de relevancia social. Para ello se pone en comparación los temas abordados por las noticias de los informativos con respecto a los temas candentes que haya en ese momento en las redes sociales.

Con objeto de que esta herramienta pueda ser utilizada de forma autónoma se la va a dotar de una interfaz gráfica que permita a un usuario usarla independientemente del resto de herramientas de la plataforma y que le proporcione toda la información que se pretende que esta parte de la plataforma ha de ofrecer.

Se pretende que el programa tenga la mayor usabilidad posible por lo que se van a implementar dos modos de funcionamiento: un modo que automáticamente con el telediario que haya disponible en ese momento y los temas de interés de ese momento realice y devuelva un análisis comparativo y otro que permita, a partir de archivos generados por la herramienta con anterioridad, realizar comparativas entre telediarios o temas de interés no actuales.

Internamente se pretende que el programa deje constancia de los diferentes procesos que ha ido realizando para obtener el resultado final, de forma que esos datos generados puedan resultar útiles para futuros análisis, contrastados con la base desde la cual se han obtenido o que se puedan consultar de forma legible por el usuario, así pues, el programa almacenará archivos con la información generada en cada una de las partes principales del análisis.

El programa en cuestión pretende:

- Crear un método simple para obtener el ultimo telediario de RTVE que se encuentre disponible a través de internet.

- A partir de un telediario en un formato de video obtener en formato texto todo el audio que contenga de la forma más precisa posible, esto es, tanto lo que digan los presentadores del informativo como lo que se diga en las diferentes noticias y almacenar esta información en un fichero de forma ordenada y estructurada.

- Obtener los diferentes temas relevantes del día de una red social y de cada uno de estos una colección de datos que resulte representativa de los temas que en ellos se traten y almacenarlos en un fichero de forma ordenada y estructurada.

- Representar cada una de estas muestras de texto en el formato ‘Bolsa De Palabras’ de forma que se permita su posterior análisis almacenando el resultado en dos ficheros distintos.

- Limpiar estas representaciones para facilitar un correcto análisis eliminando preposiciones artículos y otras palabras de poco interés para su estudio.

- Facilitar la reutilización de los datos para futuras herramientas

- Puesta en común de ambas representaciones y representación de los resultados en forma de gráficas y tablas.

- Permitir tanto un análisis automático que permita unos resultados del análisis de los datos disponibles en ese momento como uno selectivo donde se escojan las fuentes que se van a analizar.

1.3 Impacto esperado

En cuanto al impacto del proyecto de forma individual, la herramienta está destinada a un público en general para que cualquier ciudadano pueda comprobar por sí mismo la relevancia de las noticias que consume. Esto puede resultar de especial interés a periodistas o gente que se dedique al control de los medios de información pues los datos proporcionados son de especial interés para este sector. Pretende pues apoyar o dar a conocer argumentos que defiendan la manipulación realizada en los medios de información analizados, dando más peso a las acusaciones que se realicen sobre ellos.

Aunque el objetivo principal del programa es ofrecer información a la ciudadanía, también puede ser interesante su uso por los propios medios de comunicación analizados con tal de mejorar la calidad y repercusión de sus contenidos.

Si consideramos el proyecto en conjunto con la plataforma de análisis que se pretende construir, la cual va a incluir otros métodos de análisis para proporcionar un análisis del informativo de mayor grado, su objetivo no es más que reforzar el conjunto del análisis a ofrecer y ampliar la calidad y cantidad de los datos que esta pueda generar.

Además de esto se pretende facilitar la creación de nuevas herramientas de análisis a partir de los datos que se generen con esta, ya sea reaprovechando sus datos o tomando esta como modelo o base de otras aplicaciones.

1.4 Metodología

La metodología que se va a emplear para la realización de este trabajo recibe el nombre de “tubería de datos” y ha sido propuesta por la de la *School Of Data*[1].

Este metodo resulta de especial utilidad al abordar un proyecto que se basa en el manejo de datos. A su vez, permite una fácil y clara ruta a seguir de principio a fin en la realización del proyecto.

El método de “Tubería de datos” definido por la *School Of Data* cuenta con los siguientes pasos:

[1] *www.schoolofdata.org*

Definición: Se define el problema que se va a resolver, esto es desde la idea más general del proyecto hasta cada uno de los objetivos y métodos que se van a emplear, esto permite saber que datos van a ser necesarios para la realización del proyecto y cuáles no, así como averiguar si dichos datos van a ser fáciles o difíciles de conseguir.

Búsqueda: Encontrar las fuentes de datos que se van a necesitar en el proyecto y establecer que técnicas van a ser las necesarias para obtenerlas en base a su fuente o alternativas a una fuente principal potencial.

Cabe destacar en este paso que conviene asegurarse que las fuentes de donde se vaya a obtener la información permitan un uso legal de esta, tanto para su obtención, como su manipulación o análisis.

Obtención: Se trata de obtener los datos desde su lugar de origen hasta nuestro ordenador aplicando las diferentes técnicas que se hayan definido para cada fuente en la fase de búsqueda.

Verificación: Verificar que los datos obtenidos resultan de utilidad, que la fuente es fiable y que no han sido modificados en la fase de obtención a causa de una mala aplicación de las técnicas usadas

Limpieza: Ordenar y refinar los datos obtenidos para extraer exactamente aquellos que sean útiles para el proyecto con un formato de fácil localización y acceso a cada dato individual. Una buena limpieza de datos facilitará enormemente la labor de análisis puesto que evita resultados indeseados o falsos y asegura que todo lo que se analiza es relevante para el proyecto.

Análisis: Realizar la tarea para la que se han obtenido dichos datos, contrastándolos entre sí y derivando de ellos la información que se buscaba en el apartado de definición. Esta constituye pues la parte principal del proyecto y en la que se obtendrán los nuevos datos que responden al problema que ha motivado el proyecto.

Presentación: Plasmar los resultados del análisis en un formato entendible para un público ya sea general o especializado y proporcionar una serie de conclusiones extraídas de la fase de análisis.

1.5 Estructura

La estructura de este proyecto va a ser la recomendada por la ETSINF para la realización de este tipo de trabajos, primeramente, se va a realizar un análisis de la situación y el problema que concierne a este proyecto, luego se van a considerar las posibles soluciones a tratar y finalmente se va a realizar una definición e implementación de la solución fijada seguida de las pruebas correspondientes y la puesta a disposición de los usuarios de la herramienta.

Cabe destacar que en la sección de anexos se puede encontrar el código fuente desarrollado y un ejemplo de uso de la herramienta y que en los pies de página pueden encontrarse las webs a las que se hace referencia en el texto.

Al principio del documento puede verse un glosario con las palabras técnicas que resulten relevantes a este trabajo.

1.6 Colaboraciones

El proyecto como tal se ha desarrollado de forma individual por el autor. Sin embargo, se ha elaborado para que forme parte de una plataforma de mayor tamaño que ofrecerá más posibilidades de análisis de cara a un informativo. Los otros módulos de esta plataforma han sido desarrollados por otros alumnos como trabajo de final del máster en Big Data Analytics impartido por la UPV y que a fecha de la redacción de esta memoria no se han finalizado ninguna de estas otras partes de dicha plataforma.

En el caso de la herramienta que ocupa esta memoria todas sus partes han sido desarrolladas en su totalidad por el autor, sin embargo, debido a la similitud de los proyectos que compondrán la plataforma conviene mencionar que la puesta en común de las diferentes aproximaciones a los problemas a resolver y los resultados que se han ido obteniendo han resultado muy beneficiosas para el correcto desarrollo de este proyecto.

Las descripciones de las herramientas se han incluido en el Anexo III de esta memoria.

2. Estado del arte

La realización de este proyecto conlleva el manejo de tecnologías de distinta índole que actualmente se encuentran en diversos estados de desarrollo y aunque la forma en la que se abordan puede no ser la más actual cabe considerar que es porque se encuentran desarrolladas por empresas privadas y que no se dispone de un fácil acceso a estas, además del complicado proceso de aprendizaje de una tecnología tan novedosa.

No obstante, el acercamiento que se realiza a estas tecnologías resulta adecuado y suficiente para el alcance de este proyecto y proporcionan buenos resultados para el nivel de profundidad que se ha querido alcanzar.

En cuanto a trabajos anteriores en esta área, el autor no ha podido encontrar ninguna herramienta similar ya sea en español o en inglés que realice las funciones mencionadas en su totalidad, sin embargo, varias partes de las que se compone el programa sí que han sido y siguen siendo estudiadas en la actualidad y cuentan con una gran base de conocimientos sobre la que poder construir este trabajo.

2.0.1 Lenguajes de programación

Actualmente existen una gran cantidad de lenguajes de programación a escoger cuyas funciones y librerías cumplen perfectamente con las capacidades que el proyecto pretende tener, lenguajes como C++, JavaScript o Python entre otros cuentan con una gran comunidad detrás que da constante soporte a sus plataformas, tienen una gran cantidad de software que es desarrollado para estos, diferentes librerías que facilitan su uso y agilizan el desarrollo, etc.

Así pues, la elección de este ha sido por preferencia del autor y se va a usar Python 3.6 para el desarrollo de la completitud de los scripts que van a componer la aplicación, tanto por las características anteriormente mencionadas como por la certeza de que posteriores versiones van a ser compatibles con el código ya escrito, la facilidad de aprendizaje y el soporte de las diferentes librerías que se pretenden usar a lo largo del proyecto.

2.0.2 Análisis del lenguaje humano

Una de las partes principales de este proyecto constituye un análisis y comparativa del lenguaje empleado en dos ámbitos tanto de forma escrita (en forma de tweets) como de forma oral (los diferentes telediarios).

Actualmente los líderes en este sector de análisis son grandes empresas del sector tecnológico como Google, Apple o Samsung que usan principalmente el análisis del lenguaje hablado para mejorar sus asistentes de voz (7) o bien el uso de lenguaje escrito para determinar en qué tipos de anuncios o contenidos podría estar interesado el usuario de su servicio (8).

En la actualidad existen proyectos de código abierto como OpenAi[2] obtienen los mejores resultados posibles usando redes neuronales con aprendizaje no supervisado refinando posteriormente los resultados para obtener el análisis más optimo al problema a analizar. Con esto se puede entrenar un modelo que entienda el texto hasta el punto de relacionar fragmentos de distintos textos en los que se hable de un mismo tema, contestar cuestiones sobre lo analizado de forma automática, evaluación de similaridades o clasificación de textos por su contenido (9).

Estos métodos analizan elementos del lenguaje como el contexto, intencionalidad, sentimentalización, etc. Elementos que no resultan útiles de cara a los objetivos principales del programa pero que no obstante de cara a otro tipo de análisis con los datos extraídos pueden resultar útiles.

Además, estas técnicas aparte de conllevar una relativa dificultad en su aplicación resultan excesivos para el cometido del programa.

2.0.3 Transcripción de voz a texto

El *speech to text* o reconocimiento del habla consiste en la traducción del lenguaje hablado a texto de forma automatizada, para esta tarea existen diferentes métodos computacionales que se pueden aplicar y que varían en complejidad y resultados. A destacar:

Dynamic Time Wraping (DTW)

Resulta útil para reconocer palabras aisladas, consiste en parametrizar la señal acústica y compararla con otra serie de señales que se saben que se corresponden con palabras definidas, se calcula la distancia respecto a estas y la plantilla que menor distancia tenga respecto a la señal es la palabra resultante.

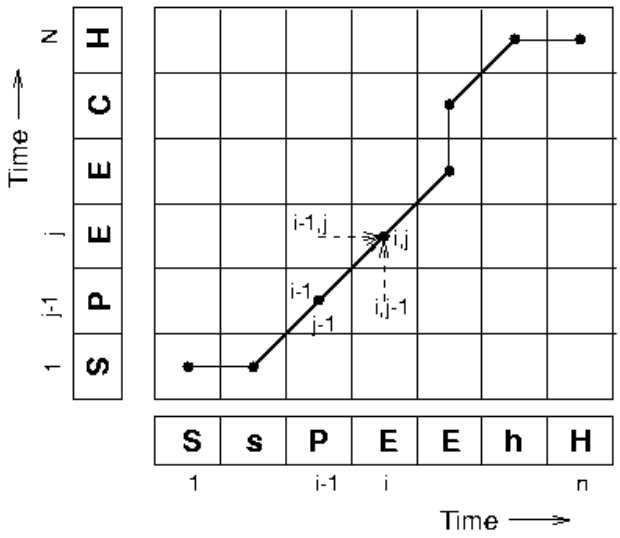


Ilustración 1 Esquema cálculo de distancias DTW

Los principales inconvenientes de esta técnica presentan graves dificultades para implementarse en el proyecto pues es necesario que la palabra esté aislada, disponer

[2] *https://blog.openai.com/language-unsupervised/*

de una base de plantillas clasificadas y que el ritmo y duración del lenguaje coincidan en ambas señales acústicas. (10)

Modelos Ocultos de Markov

Consiste en generar una máquina de estados donde cada estado representa un sonido diferente del habla, la señal después se procesa y va cambiando de estado en estado hasta llegar a un estado final que representa la palabra predicha. Esto implica restricciones temporales respecto a la palabra a analizar puesto que cada estado representa un sonido con una duración determinada que debe coincidir con cada una de las partes de la palabra. Además, al igual que en el modelo anterior se requiere que las palabras estén separadas y no se puede introducir un discurso completo a analizar ya que el modelo a generar sería de una gran extensión. (11)

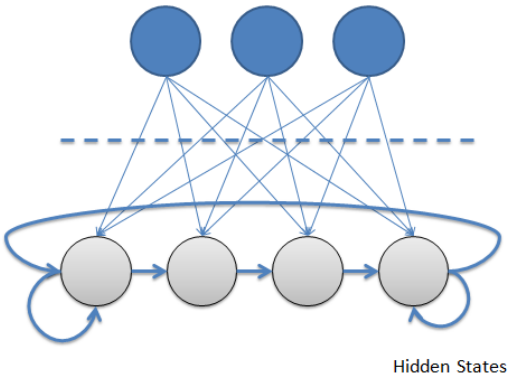


Ilustración 2 Esquema modelo oculto de Márkov

Redes Neuronales

Los modelos de conversión de voz a texto basados en redes neuronales se desarrollaron y empezaron a utilizar a finales de los años 80 pero se abandonaron debido a la falta de buenos algoritmos de entrenamiento y falta de potencia computacional.

Estos últimos años este problema ha ido desapareciendo con el creciente aumento de la potencia computacional disponible gracias al desarrollo de hardware específico para este tipo de modelos y la creación de algoritmos de entrenamiento más eficientes.

Gracias a todo esto los modelos de redes neuronales han demostrado su gran potencia computacional y eficacia en muchos ámbitos, destacando en las áreas de reconocimiento y análisis, desplazando o complementando a otras técnicas que ya existían en esos campos.

Su implementación a grandes rasgos depende del tipo de red interna que se haya diseñado y en cómo sus diferentes nodos se equilibren, pero suelen depender de tener una base de conocimiento con muestras similares a las que se pretende analizar que estén correctamente etiquetadas y clasificadas y usar esta base de conocimiento para entrenar un modelo compuesto de nodos que irán tomando distintos pesos en base a si las predicciones de clasificación que hace la maquina son correctas o no y repetir

hasta que la salida de la red comparada con la salida esperada proporcione un error mínimo.

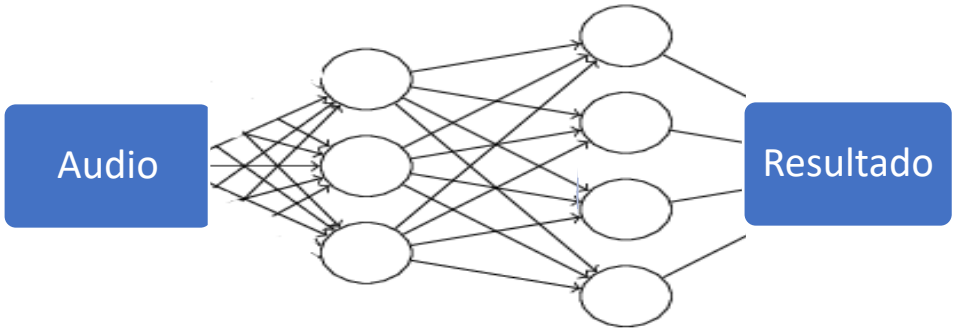


Ilustración 3 Esquema red neuronal

Debido al auge de este tipo de sistemas en los últimos años hay disponibles una buena cantidad de opciones ya sean de carácter privado como *open source* que permiten una gran adaptación al problema a tratar.

Su mayor problema radica en la complejidad de su implementación, ajuste del entrenamiento y el tener la mencionada base de entrenamiento, aunque estos problemas pueden evitarse si se usa una maquina ya entrenada para nuestro problema, que en este caso es el reconocimiento del lenguaje castellano.

Debido han surgido servicios que ofrecen el uso de esta tecnología de forma cómoda simplemente enviando el audio a analizar y devolviendo el resultado con un cierto margen de confidencia en los resultados, esto permite un uso fácil de esta tecnología al proporcionar una interfaz de interacción más simple con la red y el no tener que preocuparse de entrenarla y configurarla correctamente.

Este método es el usado por los principales asistentes de voz que se encuentran en los smartphones de última generación ya que es el que ofrece mejores resultados y en el que las empresas privadas más esfuerzos están dando. Existen también opciones de código abierto como librerías ya listas para realizar una implementación lo más sencilla posible de una red neuronal.

En cuanto a la obtención de los data-sets para entrenamiento existen bibliotecas online[3] que proporcionan de forma gratuita colecciones de todo tipo de archivos ya etiquetados y listos para su uso en tareas de entrenamiento de redes neuronales o para los otros métodos mencionados con anterioridad

En general se puede extraer que en el campo del reconocimiento del habla queda aún mucho por investigar, ya que no existe todavía un reconocedor de propósito general capaz de transcribir un discurso en cualquier lenguaje con garantías de exactitud y, aunque las empresas privadas están dando grandes pasos hacia ello (7), no existe una solución de código abierto que se encuentre próxima a solucionar este problema. Se deduce pues que el estado de la tecnología para realizar este proyecto es suficiente ya

[3] *http://deeplearning.net/datasets/*

que se ofrecen gran cantidad de opciones, algunas de las cuales de fácil uso y buenos resultados.

2.0.4 API de terceros

Las API o interfaz de programación de aplicaciones son una serie de rutinas protocolos y herramientas usadas para construir aplicaciones software definiendo como los componentes deben interactuar.

En la actualidad existen diferentes tipos de API que permiten interactuar con toda clase de sistemas, servicios o webs lo que permite a los desarrolladores programar aplicaciones con mayor rapidez y con optima interacción con ese servicio.

Ejemplos como las API de *Google Maps* o Twitter muestran el desarrollo y optimización de este tipo de servicios ofreciendo interfaces en forma de librerías para diferentes lenguajes de programación las cuales unifican internamente el tipo de llamada a sus servidores devolviendo un mismo tipo de datos independientemente del lenguaje desde el que se haya efectuado la petición.

Actualmente estos servicios, sobre todo si se trata de servicios de empresas privadas, se ofrecen de forma gratuita con ciertas limitaciones y en caso de requerir mayor potencia o tener una gran cantidad de peticiones se aplican planes de pago dependiendo del tipo de servicio y la cantidad de peticiones.

2.0.5 Redes sociales

Las redes sociales han pasado a formar una parte importante de la vida de muchas personas, permitiendo la comunicación de información a grandes distancias y a una gran cantidad de gente de forma sencilla. Esto provoca que los usuarios de estas plataformas tiendan a dar su opinión sobre temas que sean de su interés o simplemente compartirlos con sus amigos o seguidores.

Con esto se generan una gran cantidad de datos individualizados de cada usuario cosa que aprovechan los propietarios de las plataformas para hacer negocio, ya sea con anuncios que se adapten a los temas de interés del usuario o comercializando sus datos personales de alguna forma (12).

Actualmente la mayoría de las redes sociales cuentan con API’s que permiten a los desarrolladores interactuar con los datos que en ellas se generan y ofrecen y a los propietarios facilitar la monetización de estos datos.

2.1 Crítica al estado del arte

Dentro de los trabajos que se han presentado en la escuela de informática de la UPV no se han encontrado trabajos que hayan tratado el tema del análisis de telediarios con el enfoque que se le ha dado en este proyecto, por tanto, con este trabajo se cubre un vacío en lo que a temas de análisis de datos provenientes de medios de información se refiere.

Como se ha mostrado en esta sección la tecnología existente resulta mas que suficiente para la implementación de un proyecto de las características que se proponen. El hecho de que no existan herramientas similares resulta extraño debido a esto ya que un mundo cada vez más digital en el que todo se mide y controla no existe cierto control sobre los medios de comunicación públicos y la información que estos ofrecen. Podemos decir pues que en el campo del análisis de información existen una falta de herramientas especializadas en esto a pesar de la creciente necesidad de estas.

2.2 Propuesta

Lo que se propone con este proyecto es la composición de un programa que haga uso de las últimas tecnologías en la conversión de voz a texto y se va a implementar de forma que sea escalable y reusable. Se compone principalmente de tres aspectos clave:

- Recolección de datos de distintas fuentes.

- Unificación del formato y limpieza de los datos.

- Comparación de datos.

Todo esto se aplica al campo del análisis informativo, más concretamente al análisis de informativos de televisión un área de conocimiento que hasta el momento no cuenta con herramientas tecnológicas que permitan su automatización o estudio objetivo.

Al no existir precedentes de herramientas similares, se puede afirmar pues que soluciona un problema o cubre una necesidad actual a la que hasta el momento no se le había dado solución directa con una herramienta especializada y constituye una base de la que partir de cara a futuros proyectos de similar índole.

3. Análisis del problema

El problema radica como ya se especifica en el titulo del proyecto en averiguar la relevancia social de las noticias de los informativos de RTVE, para ello se va a crear una herramienta que recoja datos de alguna red social y los compare con los generados por los informativos. Para ello se deben determinar las fuentes de estos datos, los métodos de obtención y los métodos de análisis a realizar.

3.1 Análisis de la seguridad

Debido a las características de la solución desarrollada no es probable que se puedan ocasionar vectores de ataque hacia el usuario, más allá de posibles *crasheos* que pueda tener la aplicación. Aun así, cabe tener en cuenta que, al obtenerse datos de internet, aunque provengan de fuentes fiables, el usuario deberá comprobar los ficheros descargados para asegurarse de que efectivamente son seguros para su uso, nada más allá de las precauciones básicas al obtener datos de terceros a través de internet.

3.2 Análisis de eficiencia algorítmica

El programa realiza la mayoría de las operaciones internas a una velocidad relativamente rápida, sin embargo, en las fases en las que se realiza una petición a servidores externos se puede notar una ralentización debido al coste computacional que suponen las operaciones solicitadas o a las latencias de respuesta del servidor.

La opción más eficiente pues sería paralelizar las partes en las que se realicen peticiones externas para agilizar el coste temporal del análisis.

Sin embargo, después de las pruebas realizadas se ha comprobado que el mayor cuello de botella lo supone la descarga y transcripción del telediario independientemente de la velocidad de la conexión o el tamaño del archivo del telediario, siendo el tiempo de descarga de los tweets mucho menor en comparación por lo que se ha decidido una estructura secuencial de ejecución.

3.3 Análisis del marco legal y ético

El marco legal que engloba la aplicación es principalmente el establecido por las fuentes de las que toma sus datos, esto es los informativos de RTVE y los tweets de los usuarios de la red social Twitter.

En el caso de los informativos, “RTVE no comercializa sus contenidos a particulares, pero sí están disponibles para su consulta para trabajos de investigación y docencia. Televisión Española también facilita ciertos contenidos a instituciones con fines docentes.” [4]

[4] *http://www.rtve.es/faqs/index.php?categoria=41*

Y para el caso de Twitter el hecho de utilizar su propia API y no recabar o comercializar datos de los usuarios de los que se obtienen los tuits hace que no suponga ningún problema legal.[5]

3.3.1 Propiedad intelectual

Una vez finalizado el proyecto este va a quedar sujeto a una licencia proporcionada por el MIT que permite copia y redistribución de la totalidad del proyecto siempre que vaya acompañado de una cita al autor.

El objetivo de esta licencia es facilitar una posible reutilización del código escrito o una ampliación del proyecto que mejore sus características.

Esta licencia queda presente en el repositorio del proyecto bajo el archivo LICENSE.

3.3.2 Otros aspectos legales

En cuanto a los datos generados por la propia aplicación, no quedan suscritos bajo ninguna medida legal y queda en manos del usuario la responsabilidad de su veracidad y uso de forma correcta.

3.3.3 Ética

Realizar una herramienta que permita un análisis objetivo sobre el trabajo de otras personas puede tener efectos negativos sobre estas pudiendo llegar a ocasionar la pérdida de su trabajo o demás perdidas económicas, no obstante, al tratarse del análisis de un servicio público que debe dar siempre el mejor servicio a los ciudadanos el autor no ha encontrado ningún problema moral al realizar este proyecto.

Queda pues en manos del usuario el uso y consecuencia que se le dé a los datos obtenidos por la herramienta.

3.4 Análisis de riesgos

El proyecto que se va a desarrollar está orientado a un usuario interesado en el campo del análisis de la información por lo que se puede prever que la base de usuarios será relativamente reducida respecto a un programa orientado al público en general. Uno de los mayores problemas que se pueden encontrar es el no reconocimiento de la necesidad del programa debido a una falta de precedentes en este tipo de estudios, lo que deja en manos de los expertos la utilidad o conveniencia de utilización de la herramienta, al menos hasta que se encuentre integrada en la plataforma de análisis lo cual se prevé que dotará de mayor contexto a estos datos y hará que resulte de mayor interés además del hecho de formar parte de un proyecto de mayor tamaño y utilidad lo que ayuda a dar visibilidad de cara a los profesionales del sector.

En cuanto a la integración con la plataforma de análisis mencionada anteriormente se prevén varios posibles riesgos de integración, a destacar:

---

[5] *https://developer.twitter.com/en/developer-terms/agreement-and-policy.html*

- Diferencias en las tecnologías usadas, esto es, en caso de que otros módulos de la plataforma necesitasen hacer uso de tecnologías comunes a esta aplicación, pero las implementen con plataformas diferentes a las usadas.

- Diferencias en la presentación de los datos al usuario, en este caso solo vamos a ofrecer resultados de un tipo de análisis, pero al funcionar en conjunción con otros módulos lo más optimo resultaría mostrar resultados de distintos análisis combinados en uno solo.

- Reutilización de datos en caso de que otro de los módulos usara los mismos datos o parte de estos para realizar sus análisis lo más optimo seria que se obtuvieran de una fuente común a ambos para no incurrir en redundancias.

- Diferencias en los tiempos de presentación de los resultados, es posible que los tiempos de análisis varíen de un módulo a otro y que algunos terminen mucho antes que otros.

Para intentar mitigar el impacto de todos estos problemas se va a diseñar la aplicación de forma modular, separándola en diferentes partes o *scripts* que realizan funciones concretas dentro de esta, de forma que en caso de necesitarse solo sería necesario modificar o sustituir una de estas partes.

Además, para facilitar la integración i la reutilización de los datos generados estos se van a almacenar en formatos ampliamente soportados y que permiten la lectura de sus contenidos por el usuario como son el JSON y el TXT, esto va a permitir que las otras partes de la plataforma puedan reutilizar estos datos en caso de necesitarlo.

3.5 Identificación y análisis de soluciones posibles

Para la realización del proyecto cabe concretar 4 puntos importantes:

Fuentes de datos

Los datos de relevancia a obtener deben provenir de una red social, a considerar dos opciones:

- Facebook: Sus usuarios comparten estados y noticias que resultan de su interés y cuenta con la mayor base de usuarios activos en este momento. También da acceso a diferentes API que ayudan a interactuar con los datos que contiene.
- Twitter: Permite saber con facilidad cuales son los temas de los que más se está hablando en cada momento ya que los unifica en sus llamados *TrendingTopics* y permite ver las ultimas opiniones de estos temas que tienen sus usuarios. Su uso es sencillo y su base de usuarios grande y variada.

Extracción de datos

Para obtener los datos provenientes de los telediarios se pueden considerar 3 opciones:

- Pedir directamente a RTVE la publicación de los guiones de sus informativos en formato digital.
- Obtener los datos desde los subtítulos de los telediarios.

- Transcribir mediante herramientas automáticas lo que se dice en los telediarios. En el caso de esta opción cabe determinar que herramientas usar.

Obtención del telediario

En cuanto a la obtención del propio telediario hay dos opciones:

- Analizar lo que se diga en directo, durante la emisión del telediario.
- A partir de un archivo de video grabado, obtenido directamente de la web de RTVE.

Métodos de análisis

Y a la hora del análisis cabe observar dos posibles métodos:

- Entrenar una red neuronal o un clasificador que a partir de cada *TT* y cada telediario diga si se ha hablado o no de ese tema.
- Presentar los resultados al usuario de forma legible y comprensible y que este realice sus propias deducciones.

3.6 Solución propuesta

En base a todas las restricciones y tecnologías expuestas se ha decidido realizar un programa que a partir de un telediario en formato de video .mp4 permita transcribir su audio mediante un servicio externo de transcripción ofrecido por Google. Obtener desde la red social Twitter los temas relevantes del momento mediante su API oficial y una vez obtenidos ambos sets de datos compararlos mediante gráficos que presentaremos al usuario.

Se ha escogido la red social Twitter porque en su diseño base ya se anima a sus usuarios a comentar cuales son los temas que más interesan en este momento lo cual resulta en un tipo de datos más representativo para los análisis que se van a efectuar en el programa.

Para la obtención del telediario resulta más practico crear un programa que permita al usuario escoger de todos los que haya disponibles en la web y que permita un fácil acceso al último emitido contra obligar a realizar los análisis al momento. En el caso de la adaptación a la futura plataforma de análisis este punto puede cambiar em base a automatizar los análisis de los telediarios conforme se vayan emitiendo.

Para la extracción de datos del telediario es imperativo que sean lo más veraces posibles y con tal de evitar modificaciones posteriores o censuras a la hora de transmitir datos al analista se va a transcribir directamente el audio del telediario.

Para esto se van a usar los servicios de *Google Speech* que facilitan una tarea tan compleja como es la transcripción de audio sin limpieza de ruido.

Otro sistema que ha sido considerado para esto es el ofrecido por IBM llamado 'Watson'[6] que además ofrece la posibilidad de identificar al interlocutor. No obstante después de algunas pruebas se ha determinado que el de Google en este momento ofrece mayor fiabilidad de transcripción, aspecto clave para este proyecto.

Finalmente, como método de análisis se ha decidido que la representación mediante gráficos ayudará a que no existan falsos positivos y puede ofrecer al analista una información mayor a la inicialmente planteada.

El programa comparará las frecuencias de aparición de las palabras relevantes de ambos sets de datos para mostrar si en el telediario se ha hablado de cada uno de los temas de la red social.

3.7 Plan de Trabajo

Se va a seguir como plan de trabajo los pasos explicados en el apartado 1.4 de metodología. En primer lugar, una investigación de las fuentes y el contexto del trabajo, luego selección de las herramientas a usar, diseño del sistema e implementación de este. Finalmente se realizará una fase de pruebas y distribución del software.

Cabe destacar que el autor empieza la realización de este proyecto con experiencia en desarrollo Python y en *web scrapping* (13) y que para muchas de las tecnologías que se van a emplear se va a requerir cierto aprendizaje.

El aprendizaje de las tecnologías a emplear se va a realizar durante la realización de este proyecto adaptando y mejorando este conforme se vayan descubriendo las nuevas funciones.

Al final del proyecto se va a realizar un repaso de todas sus funciones para asegurarse de que se han empleado en todas sus partes las más eficientes aplicaciones de las tecnologías empleadas.

3.8 Presupuesto

El desarrollo de este programa no cuenta con un presupuesto como tal sin embargo cabe considerar que uno de los servicios externos que se van a usar en su desarrollo sí que puede inquirir costes para el usuario.

El servicio de *Google Speech* incurre en gastos si se superan más de 60 minutos de audio transcrito al día por lo que el caso de analizar varios telediarios en un mismo día supone un problema.

Las funciones de reconocimiento de voz pasados los 60 minutos diarios suponen 0,006 $ cada 15 segundos.

Cada solicitud, después de los 60 minutos diarios, se redondea al siguiente incremento de 15 segundos. Por ejemplo, 3 solicitudes independientes con 7 segundos de audio

[6] *https://www.ibm.com/watson/services/speech-to-text/*

cada una se facturarían como 45 segundos de audio (3 × 15 segundos). El importe total sería 0,018 $. Las fracciones de segundo se tienen en cuenta al redondear al siguiente incremento de 15 segundos. Es decir, 15,14 segundos se redondean hacia arriba y se facturan como 30 segundos[7].

Sin embargo, al registrar una nueva cuenta de Google se ofrecen 255,44 € de forma gratuita para poder probar sus servicios, lo que junto con el margen de 60 minutos diarios no suponen un problema para el correcto uso de la aplicación durante un periodo de tiempo considerable.

Mas allá de los gastos de una conexión a internet y del uso de un ordenador personal no existen más gastos para el uso o desarrollo de este proyecto.

[7] *https://cloud.google.com/speech-to-text/pricing*

4. Diseño de la solución

En cuanto al diseño, se ha determinado que el programa conste de cinco funciones principales que se van a presentar al usuario con tal de dotarlo de la mayor simplicidad de uso posible, además se ha diseñado una interfaz gráfica que facilita su uso para usuarios no experimentados en programas sin esta.

4.1 Arquitectura del Sistema

Como ya se ha comentado anteriormente el sistema internamente va a constar de la mayor modularidad posible de cara a posibles cambios o adaptaciones de la aplicación por lo que cada una de las cinco funciones principales a realizar está dividida en varios scripts.

El programa está organizado en un sencillo sistema de ficheros que se muestra a continuación:

```
C:./Analizador de Telediarios
|    analisis.py
|    chromedriver.exe
|    cloud_speech_to_text.py
|    config.py
|    descarga_video.py
|    extract_tweets.py
|    gcloud_snipets.py
|    limpieza_speech.py
|    limpieza_tweets.py
|    main.py
|    requirements.txt
|    tve-speech-to-text-clave.json
|    video_to_sound.py
|
|---json_speech
|
|---json_tweets
|
|---raw_speech
|
|---raw_tweets
|
|---res
|
|---sonido
|
|---videos
```

Ilustración 4 Esquema de ficheros y carpetas del proyecto

Esta estructura permite una fácil organización de los datos que se irán generando a lo largo de la ejecución del programa. El propósito de cada una de las diferentes carpetas es el siguiente:

- **Analizador de Telediarios**: Es la carpeta principal en la que están ubicados todos los componentes del programa sin incluir sus dependencias.

- **json_speech:** Ubicada dentro de la carpeta principal, almacena los archivos en formato JSON generados durante la fase de transcripción de audio a texto de los telediarios.

- **json_tweets:** Ubicada dentro de la carpeta principal, almacena los archivos en formato JSON generados durante la fase de obtención de los tuits después de ser limpiados.

- **raw_speech:** Almacena en ficheros de texto en formato TXT la trascripción del audio de los telediarios tal cual se recibe desde el sistema remoto.

- **raw_tweets:** Sirve para almacenar los tweets tal y como se reciben desde el sistema de Twitter en formato TXT.

- **res:** Contiene diferentes recursos extra que pueda usar el programa.

- **sonido:** Almacena los archivos de sonido en formato FLAC generados al separar el audio del video del telediario.

- **videos:** Almacena los telediarios descargados en formato MP4 desde la web de RTVE.

4.1.1 Descarga

La primera función del proyecto consiste en permitir la descarga del ultimo telediario que se encuentre disponible en la web de RTVE[8], esto se hará comprobando cual es el último telediario que se encuentre disponible, y presentándole al usuario la posibilidad de descargarlo o no.

El diagrama de esta función es el siguiente:

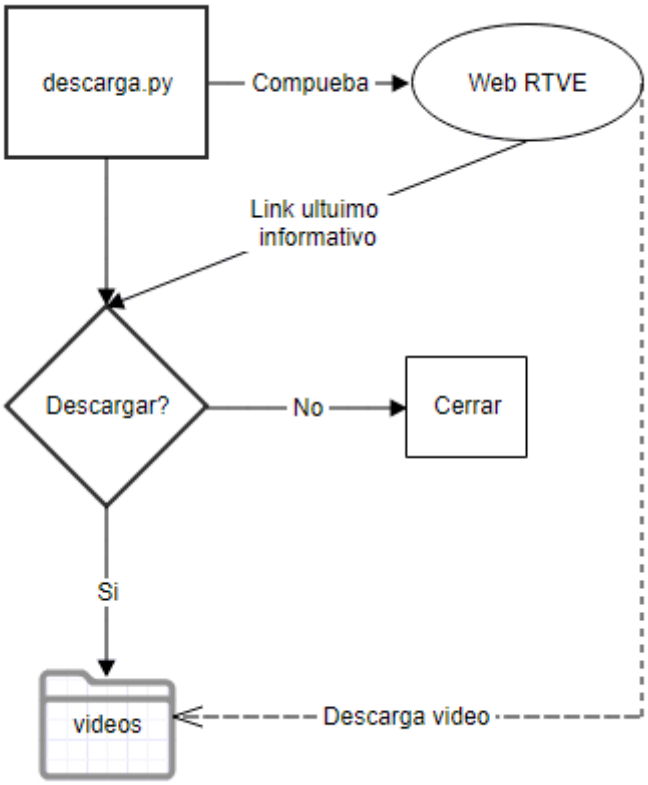


Ilustración 5 Diagrama de flujo de la función Descarga

[8] *http://www.rtve.es/alacarta/videos/telediario/*

4.1.2 Obtención de archivos a partir de Twitter

En esta función el programa obtiene mediante la API de Twitter los tweets necesarios para el análisis los limpia y los guarda automáticamente tanto en formato TXT con los tweets sin limpiar como en formato JSON con los tweets ya ordenados y limpios.

El diagrama de ejecución de esta función es el siguiente:

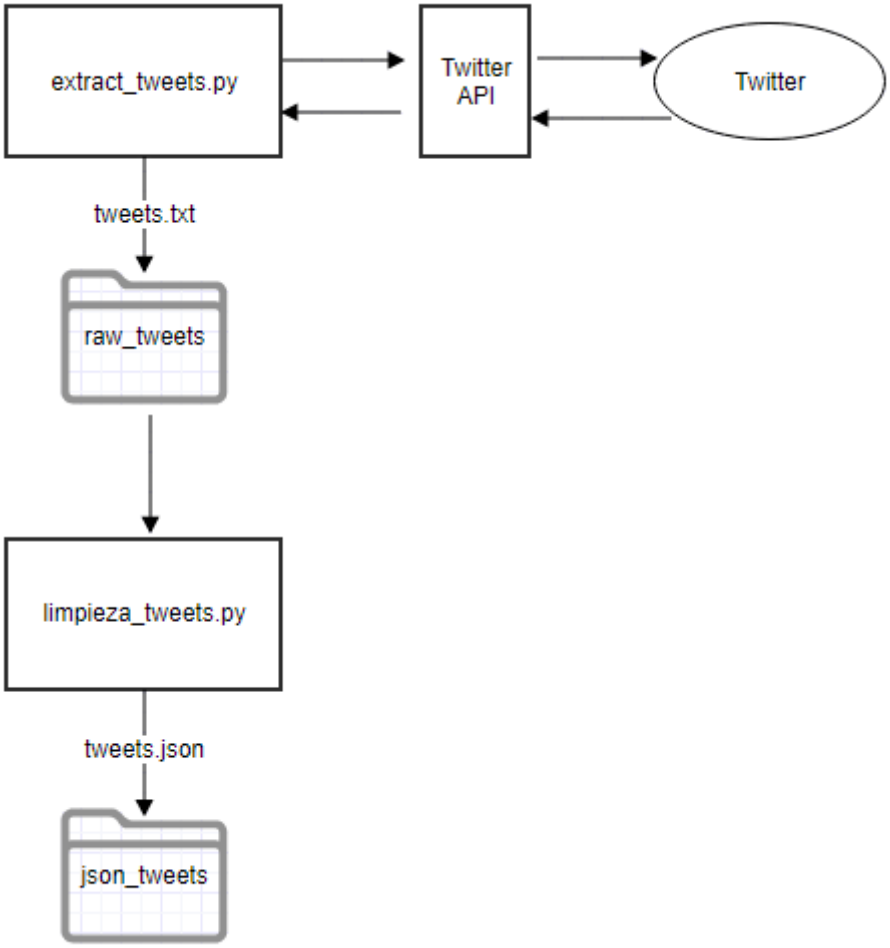


Ilustración 6 Diagrama de flujo de la función de obtención de archivos de tweets

4.1.3 Obtención de archivos del telediario

Esta característica del programa permite, a partir de un telediario descargado, generar los archivos necesarios para su análisis.

Esto se hace extrayendo el audio mediante el programa Ffmpeg y subiéndolo a los servidores de Google para su transcripción a texto. Este texto se guarda en formato TXT y luego se ordena en formato JSON

El diagrama de funcionamiento de esta parte se puede ver a continuación:

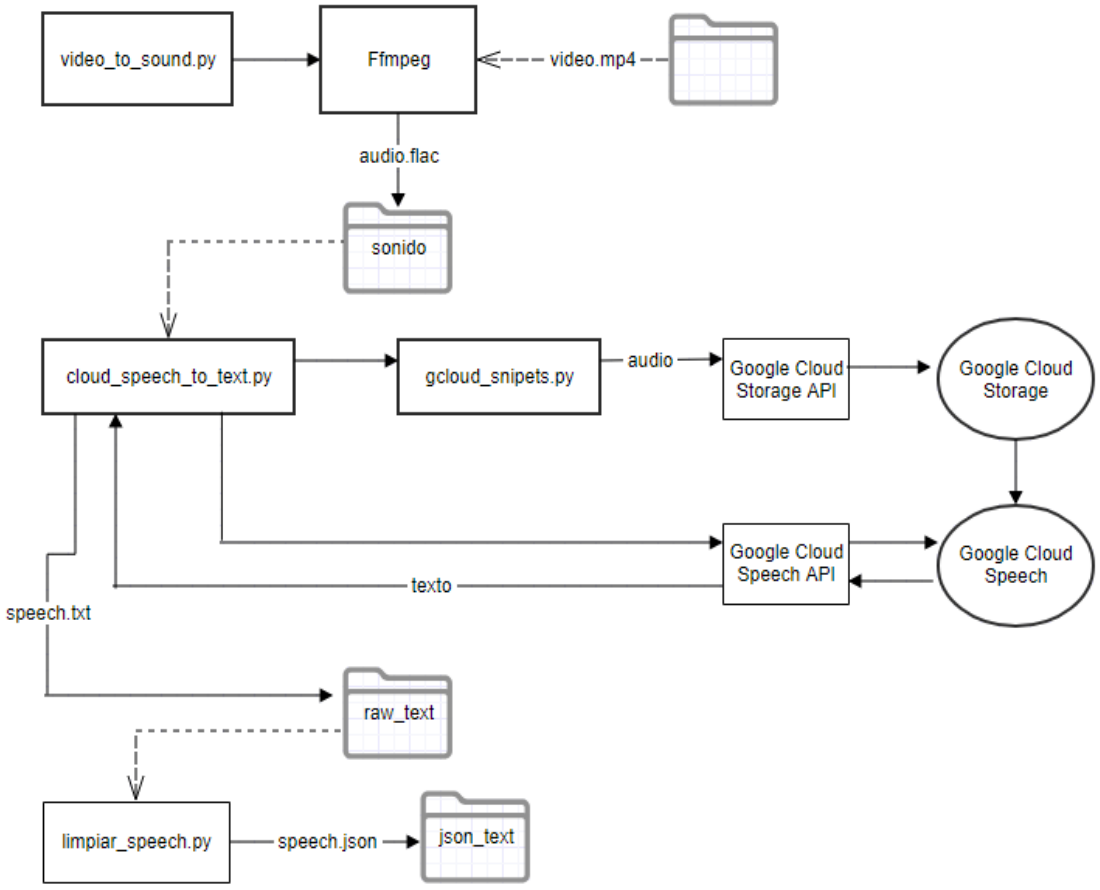


Ilustración 7 Diagrama de flujo de la función de conversión de video a texto

4.1.4 Análisis personalizado

Este modo permite al usuario realizar un análisis a partir de los ficheros generados en las funciones explicadas en los puntos 4.1.2 y 4.1.3 y que se le muestre un resultado en forma de gráficos.

El diagrama de flujo de esta función se puede representar de esta forma:

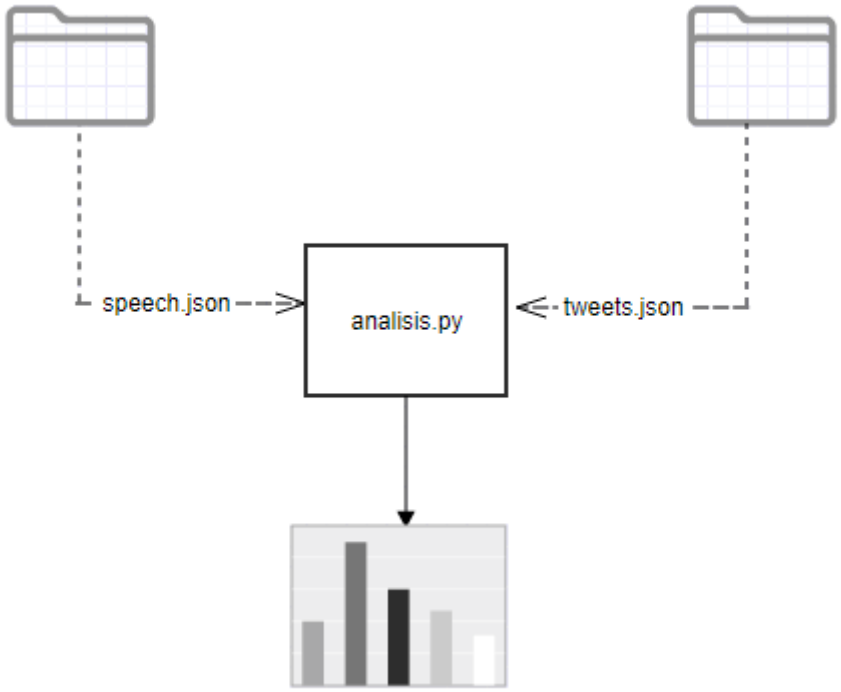


Ilustración 8 Diagrama de flujo de la función de la fase de análisis personalizado

4.1.5 Análisis automático

Este modo realiza un análisis al momento, generando todos los archivos de las fases anteriores, es decir, automatiza el proceso de realizar un análisis desde la descarga de video y obtención de tweets inicial hasta la muestra de resultados final.

El diagrama de flujo de esta función es el siguiente:

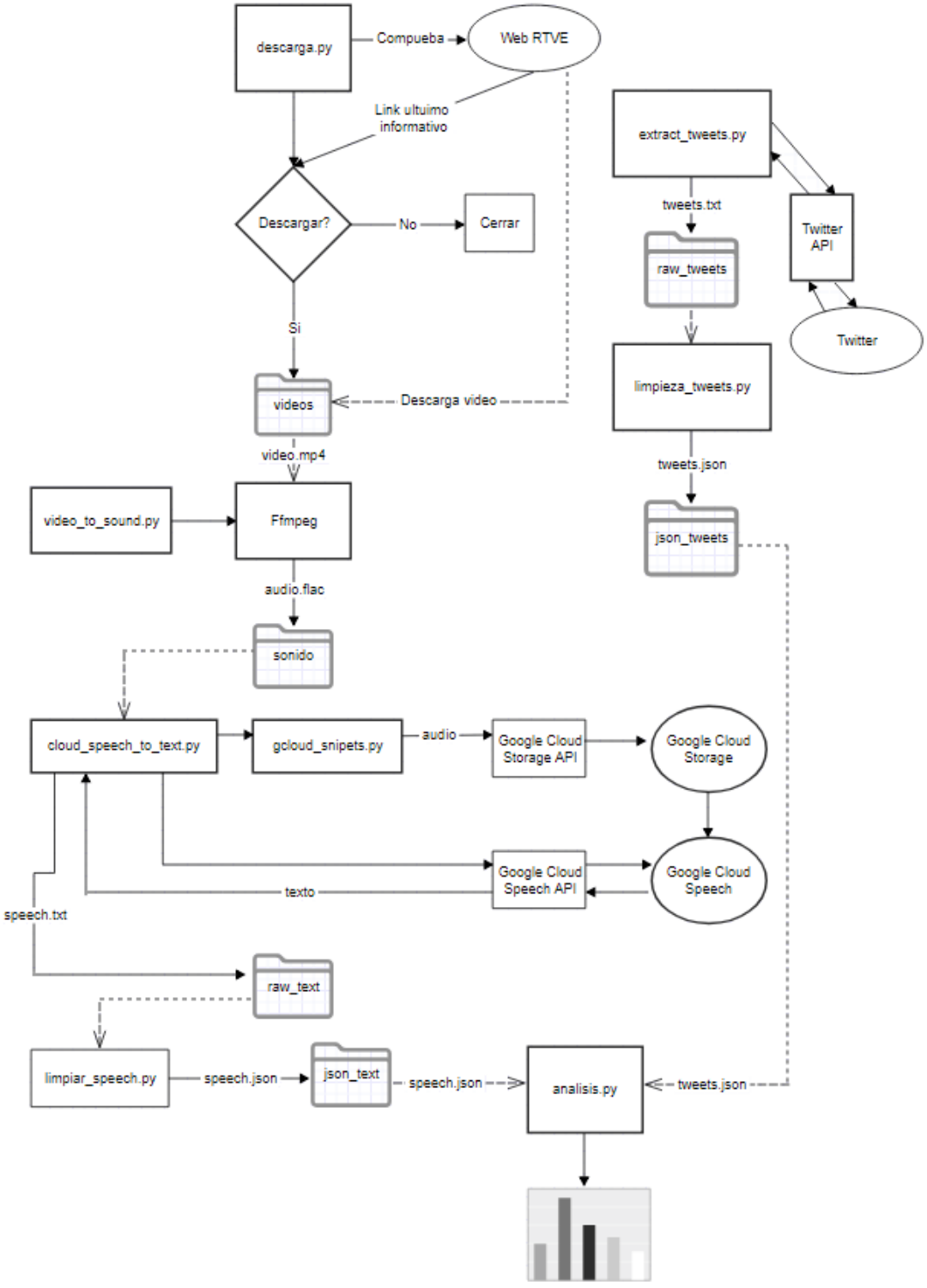


Ilustración 9 Diagrama de flujo de la función de análisis automático

4.1.6 Diseño de la interfaz gráfica

El diseño de la interfaz gráfica tiene como principal propósito proporcionar la mayor simplicidad de uso posible de cara al usuario creando ventanas con una distribución de las opciones y la información clara y concisa que permitan un fácil uso de esta.

Las diferentes ventanas que se van mostrando a lo largo de la ejecución del programa son las siguientes:

Ventana principal

Muestra los botones que permiten la ejecución de las diferentes funciones del programa.

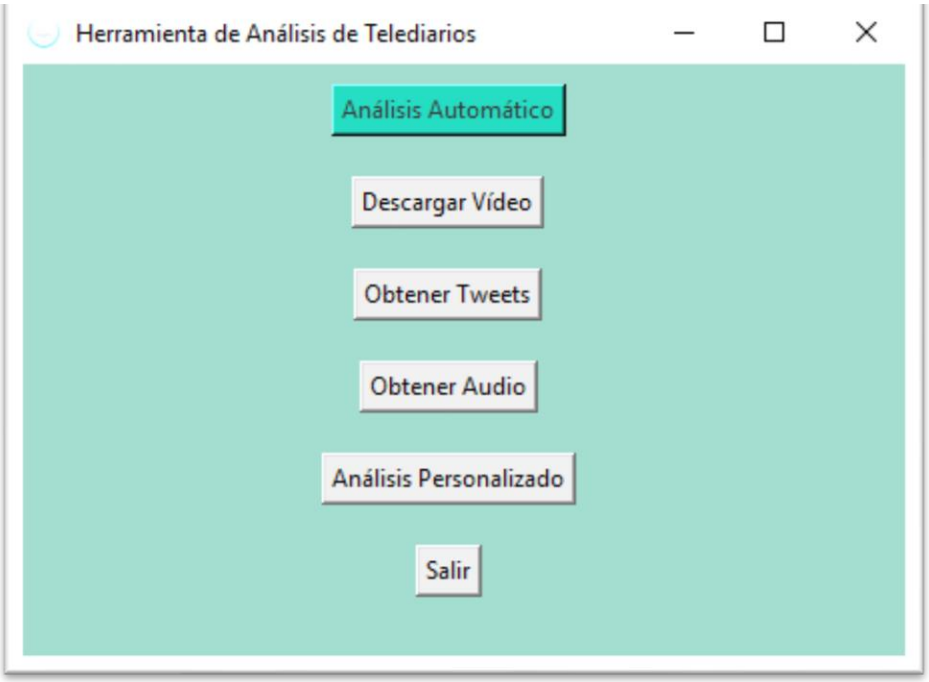


Ilustración 10 Ventana principal de la aplicación

Como se puede ver cada uno de los botones corresponde con una función de las descritas anteriormente y además se incluye un botón que cierra la aplicación.

Ventana de información

Al encontrarse la mayoría de los procedimientos automatizados muchas de las ventanas que se muestran resultan meramente informativas para el usuario indicándole en qué estado se encuentra el programa o en que procesos ha ocurrido algún error.

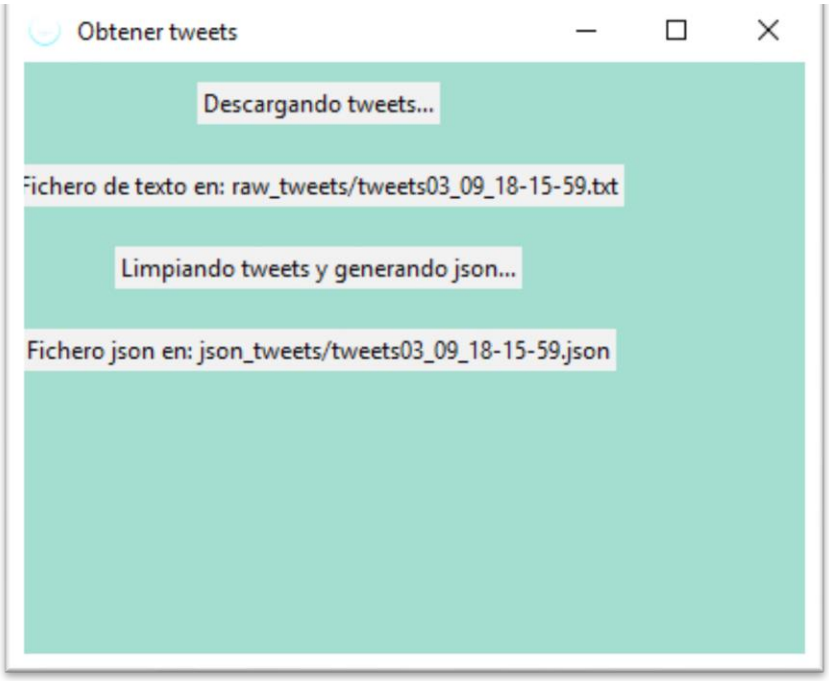


Ilustración 11 Ejemplo ventana de información

En el caso de los modos de “Análisis automático” y “Descarga” se generan dos botones en esta ventana de información para asegurarse de que el usuario efectivamente quiere descargar el telediario encontrado.

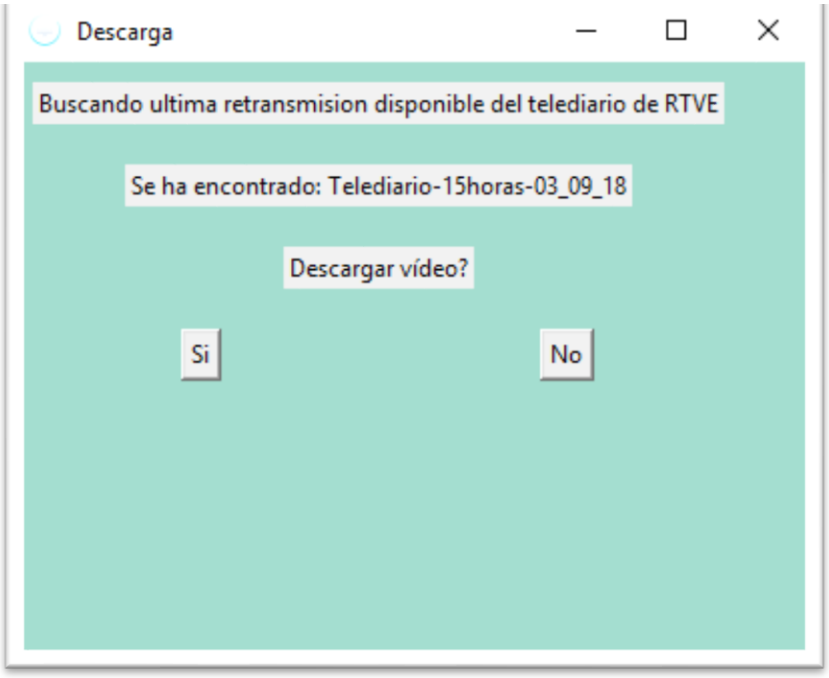


Ilustración 12 Ventana selección de descarga

Ventana de resultados

La ventana de resultados muestra cada tópico analizado junto a un botón de muestra de resultados que presenta el grafico correspondiente al análisis de dicho tópico.

Ilustración 13 Ventana de resultados

Ventana de gráficos

Al pulsar cualquiera de los botones de la ventana de resultados (Ilustración 13) se genera una ventana que va cambiando su contenido en base al resultado que se quiera ver y que muestra los resultados del análisis.

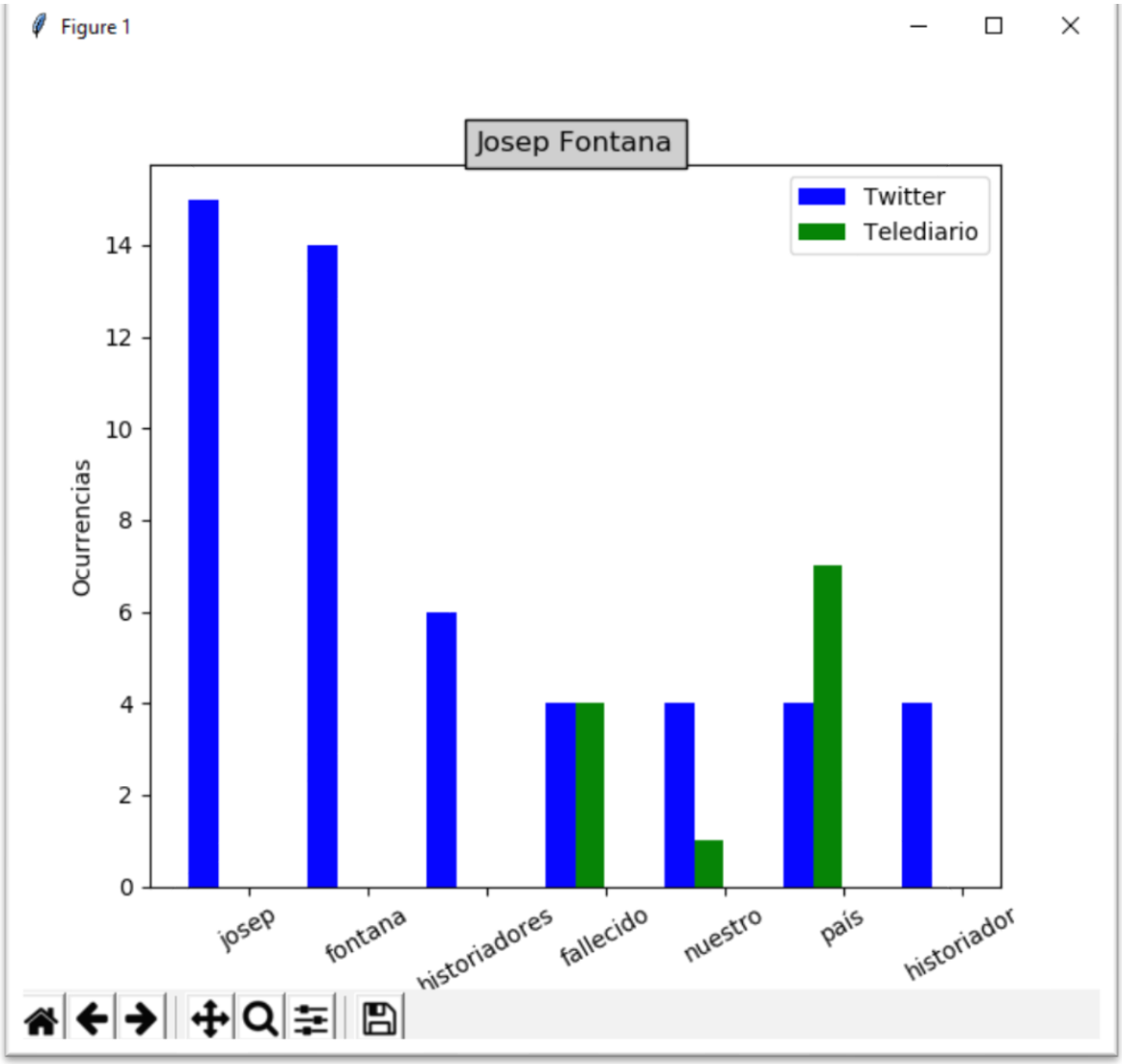


Ilustración 14 Ventana de gráficos

Esta ventana permite además guardar los resultados en formato JPG si el usuario lo necesita.

4.2 Diseño Detallado

El programa consta de diversos scripts que interactúan entre sí. El diseño interno de estos está pensado para contener las acciones que representan de la forma más eficiente posible y las estructuras de datos generadas únicamente contienen los datos necesarios para la realización de los análisis.

4.2.1 Estructuras de datos

Para almacenar las frecuencias de aparición de las palabras se van a usar dos estructuras de datos similares dependiendo de la fuente de la que se generen.

En el caso del almacenaje de los datos de transcripción de voz a texto se van a almacenar las palabras en un diccionario único con formato '*llave: valor*' donde cada palabra aparece como una llave y el valor corresponde con el número de apariciones de esta. Para dar más importancia a los nombres propios, por motivos propios del análisis,

se ha determinado que aquellas palabras que empiecen por mayúscula cuenten por tres apariciones. El formato de este diccionario puede verse a continuación:

Dic_speech = {palabra_1: nº_apariciones, …, palabra_n: nº_apariciones}

Tabla 1 Formato diccionarios de transcripciones

Para el almacenaje de las frecuencias de palabras de los tweets se ha optado por un formato similar a este, con un diccionario que recoge todos los *trending topics* ordenados por su orden de relevancia en la red social en ese momento con formato *'puesto_en_el_ranking: diccionario_frecuencias'*. El valor de cada una de estas entradas es un diccionario con las frecuencias de las palabras de ese tópico en concreto, con la forma *'palabra_n: nº_apariciones'*. El formato general de este diccionario es el siguiente:

Dic_tweets = {1: {palabra_1: nº_apariciones, …, palabra_n: nº_apariciones}, …, n: {…}}

Tabla 2 Formato diccionarios de tuits

Para permitir almacenar el tópico al que pertenece cada uno de estos diccionarios internos también contienen un elemento de llave 'TRENDING' cuyo valor es el nombre del tópico.

Estas estructuras son construidas internamente por el programa usando el tipo predefinido de Python '*dictionary'* y luego se almacenan y cargan haciendo uso de la biblioteca Json que permite una conversión directa y fácil.

Para generar estos diccionarios previamente el programa almacena los datos que recibe en formato de texto plano en un archivo TXT. En el caso de la transcripción se van escribiendo en el archivo línea a línea las diferentes respuestas que llegan desde el servicio de *Google Speech.* En el caso de los tuits, el fichero de texto contiene en una línea el tópico al que pertenecen los tuits siguientes precedido por la marca '*-#123#-*' que permite identificar donde empiezan y donde termina cada colección de tuits.

4.2.2 Estructuras de los ficheros

A continuación, se detallan las diferentes funciones que se encuentran en cada archivo *.py que conforman el programa, las funciones de las bibliotecas y otras dependencias se detallan en el apartado 4.3 de este documento:

analisis.py

Dependencias: Json, numpy, matplotlib

Funciones:

Nombre	**Parámetros**	**Return**	**Descripción**
cargar	tweets(*string*), speech(*string*)	DIC_TWEETS (*dict*) DIC_SPEECH (*dict*)	A partir de dos archivos JSON carga dos objetos de tipo diccionario con sus contenidos.

extract_trends	dict_tweets(*dict*)	lista_trends (*list*)	Extrae el termino TRENDING de los diccionarios de tuits y devuelve una lista ordenada de estos.
genera_tabla	trend (*list*) dic_tweets (*dict*) dic_speech (*dict*)		Genera el grafico de análisis a partir de un tópico, su diccionario de frecuencias y el diccionario de frecuencias del telediario.

Tabla 3 Funciones analisis.py

Estas funciones son manejadas desde *main.py* para la generación de los gráficos de resultados.

cloud_speech_to_text.py

Dependencias: pathlib, npath, google.cloud , google.cloud.speech

Funciones:

Nombre	Parámetros	Return	Descripción
subir_sonido	ruta (*string*)	url *(string),* original_uri(*string*)	A partir de la ruta a un archivo de sonido lo sube al servicio de *Google Cloud* y devuelve la ruta relativa del servicio y un enlace directo al archivo.
transcribe_gcs	gcs_uri (*string*)	nombre_fichero (*string*), ruta (*string*)	A partir de una ruta relativa de *Google Cloud* se comunica con el servicio de *Google Speech* para transcribirlo y guarda la respuesta en un archivo TXT.

Tabla 4 Funciones de cloud_speech_to_text.py

Estas funciones son manejadas desde *main.py* para la transcripción de un archivo de audio a texto.

config.py

Dependencias: os

Este archivo es usado para la fácil configuración de algunos aspectos del programa y debe ser configurado antes del uso de este. Las variables que contiene y su uso son las siguientes:

Variable	Tipo	Descripción
CONSUMER_KEY	*string*	Una de las claves necesarias para el uso de la API de Twitter.
CONSUMER_SECRET	*string*	Una de las claves necesarias para el uso de la API de Twitter.

OAUTH_TOKEN	*string*	Una de las claves necesarias para el uso de la API de Twitter.
OAUTH_TOKEN_SECRET	*string*	Una de las claves necesarias para el uso de la API de Twitter.
NUMERO_TRENDS	*integer*	Indica los N primeros tópicos de Twitter a obtener.
NUMERO_TWEETS	*integer*	Indica cuantos tuits por tópico deben obtenerse.
TIPO_RESULTADO	*string*	Modifica el tipo de tuits devueltos por la API, son posibles 3 opciones: '*popular*': Tuits más relevantes. '*recent':* Ultimos tuits disponibles. 'mixed': Una mezcla de ambos.
PALABRAS_GRAFICO	*integer*	Numero de palabras que aparecen en los gráficos de análisis.
FILTRO	*list*	Lista de palabras (*strings*) a filtrar en la generación de ambos tipos de ficheros JSON.

Tabla 5 Variables de configuración de config.py

descarga_video.py

Dependencias: selenium, time, requests.

Funciones:

Nombre	Parámetros	Return	Descripción
comprobar		link (*string*), fecha (*string*)	Averigua cual es el último telediario de RTVE disponible online y devuelve su enlace de descarga.
descarga_vid	link (*string*), name (*string*)	name (*string*)	A partir de un enlace al un video en formato MP4 lo descarga en la carpeta videos y devuelve su ubicación.

Tabla 6 Funciones de descarga_video.py

Estas funciones son manejadas desde *main.py* para realizar la comprobación y descarga de los distintos telediarios que se quieran descargar.

extract_tweets.py

Dependencias: sys, time, twitter.

Funciones:

Nombre	Parámetros	Return	Descripción
extraer_tweets	ruta (*string*)	nombre_fichero (*string*) ruta (*string*)	Obtiene los tópicos y tuits actuales mediante la API

			de Twitter y los ordena en un fichero TXT.

Tabla 7 Funciones de extract_tweets.py

Esta función es manejada desde *main.py* para la obtención de los ficheros TXT de tuits para su posterior limpieza.

gcloud_snipets.py

Este fichero contiene una serie de funciones que facilitan la interacción con el servicio de *Google Cloud Storage* y ha sido proporcionado por Google, Inc. bajo una licencia Apache 2.0 [9].

El programa solamente usa algunas de las funciones que contiene que se definen a continuación:

Dependencias: datetime, pprint, google.cloud.storage.

Funciones:

Nombre	Parámetros	Return	Descripción
create_bucket	bucket_name *(string)*		Crea un nuevo *bucket* en el servicio.
upload_blob	bucket_name *(string)* source_file_name (*string*) destination_blob_name (*string*)		Sube un archivo al *bucket* indicado con un nombre *blob* indicado.
generate_signed_url	bucket_name *(string)* blob_name (*string*)		Genera un enlace directo a un archivo subido.

Tabla 8 Funciones gcloud_snipets.py

Estas funciones son manejadas desde *cloud_speech_to_text.py* para la subida de los archivos de audio a la nube de Google.

limpieza_speech.py

Dependencias: re, json.

Funciones:

Nombre	Parámetros	Return	Descripción
limpiar_speech	nombre (*string*) ruta (*string*)	jroute (*string*)	Limpia un fichero de texto, proveniente de una transcripción, de caracteres indeseados y realiza el análisis de frecuencia de las palabras restantes almacenando el resultado en un fichero JSON.

Tabla 9 Funciones limpieza_speech.py

[9] *http://www.apache.org/licenses/LICENSE-2.0*

Esta función se usa desde *main.py* para la creación de los archivos JSON derivados de las transcripciones.

limpieza_tweets.py

Dependencias: emoji, re, json

Funciones:

Nombre	Parámetros	Return	Descripción
limpiar	nombre (*string*) ruta (*string*)	jroute (*string*)	Limpia de emoticonos y palabras indeseadas los tuits almacenados en un archivo TXT y realiza el análisis de frecuencia de las palabras restantes almacenando el resultado en un fichero JSON.

Tabla 10 Funciones de limpieza_tweets.py

Esta función se usa desde *main.py* para la creación de los archivos JSON derivados de la obtención de los tópicos y los tuits correspondientes.

video_to_sound.py

Dependencias: os, npath.

Funciones:

Nombre	Parámetros	Return	Descripción
extraer_sonido	ruta (*string*)	nombre_archivo (*string*)	Separa el audio del video y ubica el archivo resultante en la carpeta de sonido.

Tabla 11 Funciones video_to_sound.py

Este archivo es el encargado de gestionar la extracción del audio del video a analizar.

main.py

Dependencias: tkinter, time, npath.

Funciones:

Nombre	Parámetros	Return	Descripción
analisis_auto			Gestiona la ejecución de la función del programa de análisis automático.
descarga_video			Gestiona la ejecución de la función del programa de descarga de telediarios.
obtener_tweets			Gestiona la ejecución de la función del programa de la obtención de tuits y la generación de todos los archivos derivados de ello.

obtener_audio			Gestiona la ejecución de la función del programa de la obtención de la transcripción y la generación de todos los archivos derivados de ello.
analisis_personalizado			Gestiona la ejecución de la función del programa de análisis personalizado.
exit			Gestiona el botón de salida de la ventana principal.
main			Función de inicio del programa, gestiona la creación de la ventana principal con todos sus botones.

Tabla 12 Funciones main.py

Este archivo es el principal de la aplicación desde el que se gestionan todas sus partes.

4.3 Tecnología Utilizada

El lenguaje de programación utilizado para el desarrollo de este proyecto es Python 3.6 sobre un sistema operativo Windows 10 haciendo uso del entorno de desarrollo PyCharm Community Edition 2018.1.4 x64. Al ser Python un lenguaje interpretado no existe inconveniente en la ejecución de este código en otros sistemas operativo, aunque cabe mencionar que los programas externos de los que hace uso pueden ser un problema para esto.

El programa cuenta con una serie de dependencias que facilitan su desarrollo y optimizan su ejecución.

4.3.1 Librerías

El programa hace uso de una serie de librerías para realizar distintas funciones en sus diferentes archivos, las librerías y sus diferentes funciones se detallan a continuación:

Json

Librería estándar de Python para el manejo de datos en formato Json que permite la conversión directa del tipo de datos interno de Python ‘d*ictionary’* a este formato. Se usa en los archivos de limpieza para guardar los datos en este formato y en el archivo de análisis para la carga de los datos a analizar.

Numpy

Librería de cálculo matemático de Python. Se usa para la ordenación de los datos a la hora de generar las tablas de análisis.

Matplotib

Biblioteca de trazado 2D de Python que produce graficas de calidad de publicación en una variedad de formatos impresos. Usada para la generación de los gráficos de análisis.

Emoji

Contiene todo el set de emoticonos definido por la asociación Unicode. Usada en la limpieza de los tuits.

Re

Este módulo proporciona operaciones de coincidencia de expresiones regulares. Se usa para la limpieza de las palabras tanto de la transcripción como de los tuits.

Os y Sys

Librerías estándar para la interacción con elementos del sistema operativo. Usadas en distintas llamadas a funciones del sistema.

Pathlib y Ntpath

Librerías para el manejo de rutas y ubicaciones de archivos. Se usan para la localización y creación de las ubicaciones de los archivos generados y usados.

Google.cloud.storage

Librerías de interacción con la API de almacenamiento de datos en la nube de Google. Se usa para la subida de los archivos de audio a transcribir a los servidores de Google.

Google.cloud.speech

Librerías de interacción con la API de transcripciones de audio a texto de Google. Se usa para la transcripción de los audios provenientes de los telediarios.

Twitter

Librería para realizar peticiones a la plataforma de Twitter e interactuar con su API. Usada en la obtención de los tweets y tópicos.

Time y Datetime

Librerías estándar de manejo de fechas y horas. Usadas para marcar los archivos con la fecha y la hora a la que se han obtenido.

Selenium

Librería usada para automatizar interacciones con el navegador web. Se usa en la búsqueda y descarga de los telediarios de RTVE.

Tkinter

Biblioteca estándar de Python para la interacción con el *Tk GUI toolkit* herramienta para la creación de interfaces graficas disponible y compatible en la mayoría de los sistemas operativos. Se usa para la creación de la interfaz del programa.

Para facilitar la instalación de todas estas bibliotecas se va a facilitar un archivo ‘requirements.txt’ que permita instalarlas de forma fácil y en su versión correcta.

4.3.2 Programas externos

El sistema hace uso de dos programas externos que deben estar en la misma máquina que la aplicación. Los programas usados son:

ChromeDriver

Ejecutable tipo plugin usado por la librería Selenium para la obtención del archivo del telediario.

Permite ejecutar acciones como si de un navegador se tratase sin la necesidad de mostrar una interfaz utilizando la tecnología WebDriver.

WebDriver es una herramienta de código abierto para la prueba automatizada de aplicaciones web en muchos navegadores. Proporciona capacidades para navegar a páginas web, entradas de usuario, ejecución de JavaScript y más. ChromeDriver es un servidor independiente que implementa el protocolo de conexión de WebDriver para Chromium.

ChromeDriver está disponible para Chrome en Android y Chrome en escritorio (Mac, Linux, Windows y ChromeOS).[10]

Ffmpeg

FFmpeg una herramienta multimedia, capaz de decodificar, codificar, transmitir, filtrar y reproducir casi cualquier archivo de audio o video. Es compatible con la mayoría de los formatos existentes sin importar su procedencia. Está amparado bajo la licencia *GNU Lesser General Public License (LGPL) version 2.1*.

Es usado para extraer el audio de los archivos de video de los telediarios. El programa permite darle al archivo de salida las características necesarias para su correcto análisis por los servicios de Google. Se usa mediante una llamada al sistema como si de una ejecución por la línea de comandos se tratase.

Resulta especialmente practico para este proyecto por la velocidad a la que realiza su tarea y el que esté disponible para los sistemas operativos habituales.

[10] *http://chromedriver.chromium.org/*

5. Desarrollo de la solución propuesta

Durante el desarrollo de la solución se han seguido los esquemas vistos en los puntos 3 y 4 de esta memoria y todo el código escrito se puede ver en el Anexo I.

Primeramente, se desarrolló la función de Descarga para permitir la obtención de los telediarios. En un primer momento se pretendía realizar un *scraping* de la página web de RTVE sin embargo el enlace al archivo fuente no se encuentra en la página por lo que se ha tenido que extraer de otra web que permite obtener este enlace implementando unos simples comandos de interacción con Selenium.

Para esto usando la librería Selenium y el archivo 'chromedriver.exe' se abre una instancia del navegador Chrome invisible al usuario y se realizan las peticiones necesarias de forma transparente a este.

Para la obtención de los tuits se experimentó con varias API no oficiales, pero al final la oficial es la que ha dado mejores resultados y mayor velocidad de obtención de los datos.

En el caso de las transcripciones, el sistema de Google permite la transcripción de fragmentos cortos de audio sin registro y durante la fase de pruebas fue suficiente para seguir. Mas tarde al pretender analizar archivos más extensos se requirió el registro de una cuenta de desarrollador y remodelar esta fase debido a esto.

El segmento de subida a *Google Cloud* fue requerido por este motivo y no incurre en ningún tipo de gasto para la aplicación ya que es ofertado de forma gratuita por Google.

Este servicio requiere la creación de un lugar de almacenaje llamado *bucket* que contendrá todos los archivos que se vayan subiendo. Para que el programa cree este contenedor una única vez, después de su primera ejecución se crea en la carpeta 'res' el archivo 'bucket_exists.txt' que está vacío. El programa comprueba la existencia de este archivo en sus sucesivas ejecuciones para no crear de nuevo el contenedor y subir los archivos directamente al que ya existe.

El archivo de audio que se analiza necesita tener unas características concretas para su correcta transcripción por lo que al programa Ffmpeg se le dan los argumentos para obtener un archivo FLAC en mono, a 1600 hercios con un flujo de datos constante. Además, el audio se limita a los primeros 59,55 segundos del video para no incurrir en gastos en caso de solamente realizar un análisis diario.

En la fase de limpieza se encontró que los datos recibidos de Twitter contenían gran cantidad de emoticonos que entorpecían la limpieza y el manejo por parte de Python de las cadenas de caracteres, esto conllevó a realizar modificaciones en el sistema para especificar el set de caracteres que usa Python y el uso de la biblioteca Emoji para eliminarlos.

Para la extracción del audio se investigó el usar bibliotecas o utilidades de Python para ello, pero al final se decidió usar un programa externo debido a las especificas características que deben tener los datos analizados por los servicios de Google.

Para el almacenamiento de los datos se plateó el uso de XML en vez de TXT, pero el resultado obtenido con TXT es suficiente y se evita complicar la lectura y escritura de los datos.

En un principio el programa no iba a contar con una interfaz gráfica y los resultados y la interfaz iba a realizarse por consola, pero con objetivo de hacerlo más atractivo de cara

al usuario, se optó por usar la biblioteca Tkinter que tiene un uso sencillo, para realizar una interfaz ligera y efectiva para este programa.

En general el desarrollo ha seguido los esquemas previstos adaptándose a los diferentes problemas que han surgido.

6. Implantación

La implantación de este proyecto constituye en la puesta a disposición del público de su código fuente y la realización de una guía de uso e instalación de sus dependencias, de forma que su obtención y uso resulten lo más sencillas posible.

Para esto se ha incluido en el repositorio donde se encuentra disponible el programa un archivo 'README.md' que contiene una versión del punto 6.2 de esta memoria para facilitar la instalación del programa a los usuarios.

6.1 Disponibilidad

Para poner a disponibilidad de los usuarios el programa se ha creado un repositorio de GitHub[11] donde se encuentran todos los archivos necesarios para su ejecución, así como la estructura de carpetas necesaria.

Dentro de cada una de estas carpetas se ha ubicado un fichero con extensión *.md para informar al usuario de su propósito.

El repositorio del programa puede encontrarse en el siguiente enlace:

https://github.com/chgiol/AnTel

6.2 Instalación

Para la correcta instalación de este proyecto se debe primeramente obtener todos sus archivos desde el repositorio en el que se encuentra. Para esto puede clonarse usando la consola de Github sin necesidad de acceder a la web o bien mediante la web con la opción 'Download Zip' lo cual requerirá la descompresión del archivo descargado.

Para la instalación de las diferentes bibliotecas de las que hace uso el programa, se ha proporcionado un fichero 'requirements.txt' que contiene todas las necesarias con sus versiones correspondientes. Para realizar la instalación de todas estas bibliotecas de forma fácil se puede usar el gestor de paquetes de Python Pip. Para esto basta con tener el gestor instalado, situarse en la carpeta donde se encuentre 'requirements.txt' y ejecutar el comando:

"pip install -r requirements.txt"

El programa requiere que el usuario registre una cuenta de desarrollo de Twitter, para ello hay que dirigirse a su página para desarrolladores[12] rellenar el formulario de registro

[11] *www.Github.com*

[12] *https://apps.twitter.com/*

y una vez completado dirigirse a la sección *Keys and Access Tokens* y copiar las llaves correspondientes en el fichero 'config.py' del programa en formato *string.*

Para el uso de los servicios de Google se requiere una cuenta de desarrollador y para esta cuenta es necesario el uso de una tarjeta de crédito. Teniendo esto en cuenta hay que dirigirse a la página de registro de desarrolladores de Google[13] y una vez efectuado el registro dar permisos a las API de *Google Cloud* y *Google Speech* desde el panel de control de la página. Luego hay que dirigirse a la sección de credenciales[14] y descargar el fichero JSON que las contiene.

La ruta a este archivo debe especificarse en el registro de variables del sistema, esto puede hacerse con el comando:

"*set GOOGLE_APPLICATION_CREDENTIALS = [ruta al archivo]"*

Si nos encontramos en un entorno Windows hay que especificar el set de caracteres de Python con el comando:

"set Python_encoding = utf-8"

Finalmente hay que descargar el programa Ffmpeg para ello hay que dirigirse a su página oficial[15] descargar el programa y descomprimir el archivo. Por último, hay que añadir la ruta del ejecutable principal de este programa a la variable 'Path' del sistema en el que se vaya a ejecutar la herramienta.

Con esto el programa está listo para usarse.

[13] *https://console.cloud.google.com*

[14] *https://console.cloud.google.com/apis/credentials*

[15] *https://www.ffmpeg.org*

7. Pruebas

Muchas de las pruebas realizadas sobre el programa se han ido efectuando siguiendo el desarrollo de este para comprobar que las diferentes partes interactuaban correctamente.

Con el programa finalizado para verificar que sus funciones funcionan correctamente se ha visionado uno de los telediarios para comprobar que noticias se mencionan y más tarde se ha comparado con los *Trending Topics* de ese día.

En dicho telediario se mencionaba la noticia del asesinato de una mujer en el pueblo de Orihuela y como puede verse, el *topic* 'Orihuela' aparece en la lista de los de ese día:

Ilustración 15 Resultados análisis 1

Al visualizar los resultados podemos ver que las principales palabras de este tema se mencionan en el telediario:

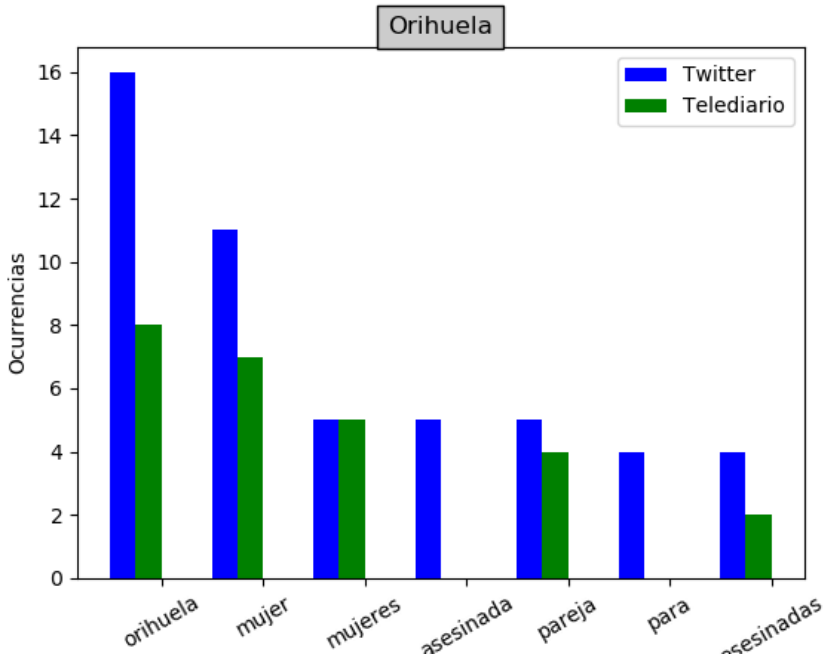


Ilustración 16 Resultados prueba 2

Esto nos indica que como efectivamente ocurre, el sistema está detectando una coincidencia en los temas tratados por lo que podemos decir que este ha funcionado correctamente.

En cambio, si vemos el grafico de un tema que no ha sido tratado:

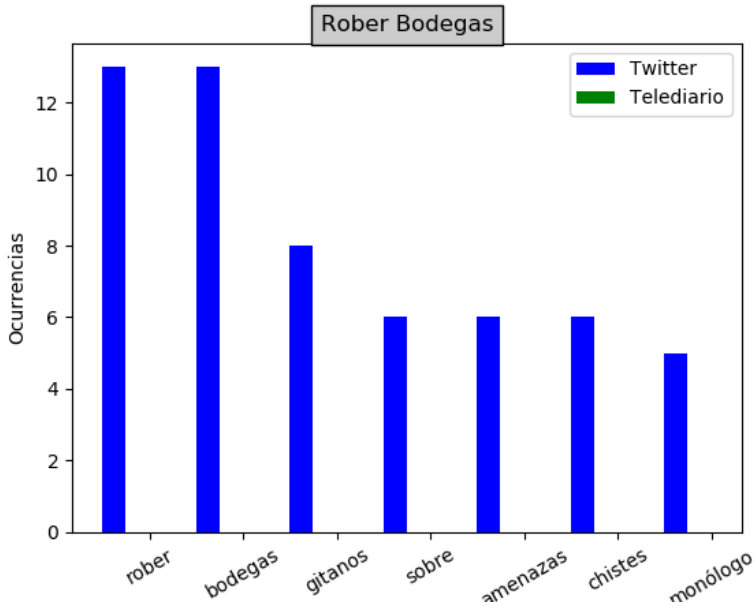


Ilustración 17 Resultado prueba 3

Vemos que en el telediario no se han mencionado en ningún momento las principales palabras de los tuits de este tópico.

Este método se ha repetido con varios telediarios y con sets de tópicos que efectivamente contenían temas tratados en ellos y no se han detectado falsos positivos obvios.

Sin embargo, el sistema en extrañas ocasiones falla en la obtención de tweets de ciertos tópicos, obteniendo menos de los solicitados, esto es, debido a limitaciones de la API de Twitter, lo cual no resulta problemático en la ejecución del programa y a la hora de visualización de los análisis es fácilmente detectable si se observa que las palabras más repetidas del tópico no aparecen un número similar de veces al número de tuits solicitados a la API.

En cuanto a la carga computacional no se considera relevante ya que las operaciones que realiza el programa internamente resultan ligeras para el estándar del hardware actual.

El tiempo de ejecución de un análisis automático pude considerarse la mayor carga para el programa ya que este realiza todas sus funciones de forma secuencial siendo las peticiones a los servicios como la descarga del video y carga del audio y transcripción de este las más costosas para el sistema.

La carga temporal del programa se ha calculado con una conexión a internet de 50Mb de descarga y 10Mb de carga con archivos audio y video de 10 y 60 minutos de duración en formatos FLAC y MP4.

Operación	**Duración archivo ~10 minutos**	**Duración archivo ~60 minutos**
Descarga video telediario	-	12 minutos
Carga a Google Cloud	1’3 minutos	6 minutos
Transcripción de Google Speech	2 minutos	16 minutos

Tabla 13 Carga temporal del sistema

Las posibles variaciones vienen dadas por los servidores externos a los que se solicita el servicio.

8. Conclusiones

En general, el programa cumple con todos los objetivos planteados en un principio y el desarrollo ha sido acorde con lo planeado.

Respecto a los objetivos propuestos en un primer momento se puede afirmar que con la inclusión del botón de descarga en el programa se ha creado un método simple para obtener el ultimo telediario de RTVE.

Si bien es verdad que cabe la posibilidad de mejora implementando un selector de los últimos telediarios, para realizar análisis generales este sistema es suficiente y además el propio usuario tiene la posibilidad de buscar el telediario a través de otras fuentes.

La obtención de la transcripción es especialmente efectiva siendo los servicios que ofrecen los sistemas de Google los más efectivos de los comprobados por el autor recibiendo transcripciones con confianzas de acierto de más del 95% en la mayoría de los casos.

Cabe a mejorar la implementación de un sistema que verifique las confianzas de estas transcripciones para mejorar la veracidad de los análisis, pero siendo los resultados notablemente buenos no se ha considerado excesivamente necesarios.

En la obtención de temas relevantes el resultado es satisfactorio, a pesar de las limitaciones de la API, sobre todo después de comprobar la ‘suciedad’ de los datos que se obtienen, se ha conseguido implementar un buen sistema que deja los datos preparados para un correcto análisis.

De nuevo a pesar de que los resultados sean suficientes un método para obtener la confianza de que los datos obtenidos a través de Twitter que son representativos resultaría un buen complemento para el programa, lo cual no se ha considerado prioritario en el desarrollo de este proyecto.

En cuanto a las representaciones hechas considero un acierto el que el programa deje los datos en texto plano y en formato ‘bolsa de palabras’ pues permite una mayor reutilización de los datos obtenidos ya sea por futuras mejoras a esta herramienta o por herramientas que trabajen en conjunción a esta en un futuro.

La limpieza de los datos ha sido notablemente más sencilla en la transcripción que en los datos de la red social, sin embargo, se ha tenido que aplicar el mismo patrón para asegurarse de la uniformidad de los datos obtenidos a la hora del análisis.

Las palabras de poco interés se han eliminado mediante longitud, eliminando las de 3 o menos caracteres del análisis, un mejor sistema resultaría mediante un filtro especifico, pero los resultados con el método actual son más que satisfactorios y existe un filtro para que el usuario pueda refinar aún más su análisis.

Los resultados de las pruebas han mostrado lo esperado en un primer momento por lo que considero este proceso exitoso ofreciendo de un vistazo la información necesaria para que el analista pueda sacar conclusiones.

Los modos de análisis funcionan como se esperaba siendo el análisis personalizado normalmente superior en efectividad debido a los tiempos en los que la población está más interesada en un tema concreto contra cuando se habla de esto en las noticias, o lo que tarda el telediario en estar disponible a través de internet.

Hay que destacar el aprendizaje de nuevas tecnologías necesario para este trabajo como la interacción con servicios web, o el uso de API de servicios privados que requieren de autenticaciones y permisos a la hora de realizar las peticiones. Estos métodos aprendidos junto con la estructuración de un programa modular o el diseño de estructuras de datos reusables van a resultar de gran utilidad en un futuro.

La gestión del tiempo disponible ha resultado ser clave durante el desarrollo debiendo de priorizar los elementos del programa claves y dejando de lado los secundarios menos importantes, para poder ofrecer un producto final dentro del tiempo.

Cabe mencionar que este proyecto es el primero al que el autor se enfrenta en solitario con un alcance y dimensiones tan amplios y que el acercamiento a este y la metodología aprendida han resultado interesantes y resultaran útiles en un futuro.

En general creo que al programa le queda margen de mejora y varias cosas se han quedado en el tintero mayormente por falta de tiempo, pero considero que el resultado final es una buena primera versión de este proyecto y una buena adición a la plataforma de análisis que se encuentra en construcción.

Relación del trabajo desarrollado con los estudios cursados

Durante la totalidad del trabajo se pueden observar las técnicas aprendidas durante los sucesivos años del grado que permiten la creación de un elemento de software funcional y usable, que permite una metodología sana a la hora de desarrollar un programa.

Las formas de abordar un proyecto aprendidas en la asignatura de ‘Soluciones TI para el sector infomediario’ o el diseño software aprendido en la asignatura de ‘Ingeniería del Software’ han resultado especialmente útiles en este proyecto.

Una de las características principales de un buen programador es la capacidad de descomponer un problema o un proyecto en elementos más pequeños a los que poder dar solución de una forma simple y en la creación de este proyecto considero que estos aspectos se han cubierto, como se puede ver en la parte de diseño de esta memoria.

El uso de diferentes tecnologías y el aprovechamiento de las características del lenguaje de programación usado para beneficiar el resultado del producto final son elementos importantes para un graduado en informática y como se puede apreciar en el proyecto en mayor o menor medida se han demostrado.

9. Trabajos futuros

El programa tiene diferentes elementos de mejora a los que con un trabajo posterior se podrían abordar para mejorar su usabilidad. Muchos de estos elementos no se han podido implementar por falta de tiempo y suponen el punto de partida a un trabajo futuro:

- Mejora de la interfaz de usuario con un aspecto más amigable para el usuario.
- Implementación de un sistema de medición de métricas de confianza del análisis realizado.
- Posibilidad de escoger entre los telediarios disponibles online.
- Mejora en las velocidades de obtención de datos de transcripción ya sea realizando esta tarea en local o realizando varias peticiones simultaneas de distintas partes del archivo.
- Mayor diversidad en las fuentes de datos de redes sociales para obtener un mayor espectro de opinión.
- Mejora en el método de distribución del programa.
- Disminución del número de dependencias del programa mediante alternativas nativas al lenguaje de programación.

Finalmente, hay que mencionar que queda pendiente la integración a la plataforma de análisis de telediarios para la que inicialmente se diseñó el programa y que se realizará cuando esta esté lista.

10. Referencias

1. **Entrevista.** Atlántica. *http://www.atlanticaxxii.com.* [En línea] 14 de Agosto de 2018. [Citado el: 03 de 9 de 2018.] http://www.atlanticaxxii.com/mujeres-rtve-caso-yolanda-alvarez/. 57.

2. **Comunicado.** vertele. *www.vertele.eldiario.es.* [En línea] 3 de 10 de 2017. [Citado el: 3 de 9 de 2018.] http://vertele.eldiario.es/noticias/Reporteros-Fronteras-periodistas-TVE-manipulacion_0_1945305457.html.

3. **Laura Uche, Alba Camazón.** eldiario. *www.eldiario.es.* [En línea] 26 de 08 de 2017. [Citado el: 3 de 9 de 2018.] https://www.eldiario.es/politica/Censura-manipulacion-constantes-informativos-TVE_0_678932159.html.

4. **Colaborativo.** Twitter. *hashtag= asisemanipula.* [En línea] [Citado el: 3 de 9 de 2018.] https://twitter.com/hashtag/asisemanipula?lang=es.

5. **RTVE, Prensa.** Rtve. *www.rtve.es.* [En línea] 22 de 06 de 2018. [Citado el: 3 de 9 de 2018.] http://www.rtve.es/rtve/20180622/jose-antonio-sanchez-concluye-su-segundo-mandato-frente-rtve-informativos-lideres-cuentas-superavit/1754324.shtml.

6. **A.G.** elPlural. *www.elplural.com.* [En línea] 26 de 6 de 2017. [Citado el: 3 de 9 de 2018.] https://www.elplural.com/comunicacion/despido-en-rtve-por-negarse-a-manipular_105739102.

7. **Kastrenakes, Jacob.** Google Assistant will soon detect what language you're speaking in. [En línea] 23 de Feb de 2018. https://www.theverge.com/2018/2/23/17041920/google-assistant-languages-multilingual-detection.

8. **Google.** Segmentar los anuncios. *Google.* [En línea] www.google.com. [Citado el: 1 de 9 de 2018.] https://support.google.com/google-ads/answer/1704368?hl=es.

9. **Radford, Alec, y otros.** *Improving Language Understanding by Generative Pre-Training.* [Paper Divulgativo] Jul de 2018.

10. **Lindasalwa Muda, Mumtaj Begam.** Voice Recognition using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques . [En línea] Marzo de 2010. https://arxiv.org/ftp/arxiv/papers/1003/1003.4083.pdf.

11. **Ling, Zhen-Hua.** *HMM-based Speech Synthesis: Fundamentals and Its Recent Advances.* s.l. : Microsoft Research, 17 de Oct de 2006.

12. **McFarlane, Greg.** *How Facebook, Twitter, Social Media Make Money From You.* [Investopedia.com] New York : Investopedia, LLC, 2016.

13. **Russell, Matthew A.** *Mining the Social Web, 2nd Edition.* s.l. : O'Reilly Media, Inc., 2013. 9781449368180.

Webs de consulta utilizadas

https://stackoverflow.com

https://schoolofdata.org

https://es.wikipedia.org

http://www.rae.es

Anexo I. Código fuente desarrollado

descarga_video.py

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import time
import requests

def comprobar():
    try:
        chrome_options = Options()
        chrome_options.add_argument("--headless")
        chrome_options.add_argument("--window-size=1280x720")

        driver = webdriver.Chrome(chrome_options=chrome_options)

        driver.get("http://www.piraminetlab.com")

driver.find_element_by_tag_name("textarea").send_keys("http://www.rtve.es/alacarta/videos/teledia
rio/")
        driver.find_element_by_id('boton_azul').click()
        time.sleep(5)
        link = driver.find_element_by_xpath("//a[@class='enlace']").get_attribute(
            "href")

        fecha = driver.find_element_by_xpath("//div[@id='info_titulo']").text
        fecha = fecha.replace(' ', '')
        fecha = fecha.replace('/', '_')
        print(link, fecha)
        driver.quit()
        return link, fecha
    except Exception as e:
        print('Error, no se ha encontrado archivo a descargar')
        driver.quit()
        return 0, 0


def descarga_vid(link='http://techslides.com/demos/sample-videos/small.mp4', name='telediario'):
    name = 'videos/' + name + ".mp4"
    print("****Connected****")
    print("Donloading.....")
    r = requests.get(link)
    f = open(name, 'wb')
    print('Writing')
    for chunk in r.iter_content(chunk_size=255):
        if chunk:  # filter out keep-alive new chunks
            f.write(chunk)
     print("Done")
    f.close()
    return name
```

```
import os

# Directorio de trabajo donde se deposita el archivo de sonido
dir_path = os.path.dirname(os.path.realpath(__file__))
'''IMPORTANTE  Si nunca has ejecutado el programa, asegurate de que en /res NO existe el archivo
bucket_exists.txt '''

# Tokens necesarios para la api de TWITTER
CONSUMER_KEY = ''
CONSUMER_SECRET = ‘’
OAUTH_TOKEN = ‘’
OAUTH_TOKEN_SECRET =‘’

# Clave para la api de google speech
#G_KEY = 'AIzaSyAHH39Z2FZyof632CGPIpSiUypEYAD9KHI'

# Configuración del archivo extract_tweets.py
NUMERO_TRENDS = 15  # Indica el numero de TT's a analizar(1-20)
NUMERO_TWEETS = 20  # Indica el numero de tweets por trend a obtener(min 1)
TIPO_RESULTADO = "popular"  # Tipo de twueets devieltos. 3 opciones: mixed recent popular

# Numero de palabras que aparecen en los graficos de analisis
PALABRAS_GRAFICO = 7
# Lista de palabras a filtrar en la generacion de ambos ficheros json
FILTRO = ['como', 'luego', 'porque', 'pues', 'pero', 'dado', 'sobre', 'para', 'esta', 'entre', 'ahora', 'sido',
        'tambien', 'hasta']

# Comando para establecer la variable de entorno del archivo .json de las credenciales de google
donde [PATH] es la ruta al archivo

# $env:GOOGLE_APPLICATION_CREDENTIALS="[PATH]"

# $env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\Christian\Documents\Python
Scripts\Proyecto TFG\tve-speech-to-text-clave.json"
```

config.py

cloud_speech_to_text.py

```
import io
import os
import config
import gcloud_snipets
from pathlib import Path
import ntpath
import time

# Imports the Google Cloud client library
from google.cloud import speech
from google.cloud.speech import enums
from google.cloud.speech import types


# Instantiates a client
# client = speech.SpeechClient()

def subir_sonido(ruta):
    my_file = Path("res/bucket_exists.txt")
    nombre_archivo = ntpath.basename(ruta)

    if (my_file.is_file()):
        gcloud_snipets.upload_blob('tve-bucket', ruta, nombre_archivo[:-5])

        url = gcloud_snipets.generate_signed_url('tve-bucket', nombre_archivo[:-5])
        original_uri = 'gs://tve-bucket/' + nombre_archivo[:-5]

        # gcloud_snipets.list_blobs('tve-bucket')
    else:
        # Creamos el bucket
        gcloud_snipets.create_bucket('tve-bucket')
        # Creamos el archivo para no volver a crear el bucket
        aux = open("res/bucket_exists.txt", "w+")
        aux.close()


        gcloud_snipets.upload_blob('tve-bucket', ruta, nombre_archivo[:-5])

        url = gcloud_snipets.generate_signed_url('tve-bucket', nombre_archivo[:-5])
        original_uri='gs://tve-bucket/'+nombre_archivo[:-5]

        # gcloud_snipets.list_blobs('tve-bucket')
    return url, original_uri


def transcribe_gcs(gcs_uri='gs://tve-bucket/Telediario-21horas-25_08_18'):
    """Asynchronously transcribes the audio file specified by the gcs_uri."""
    from google.cloud import speech
    from google.cloud.speech import enums
    from google.cloud.speech import types

    # Instantiates a client
    client = speech.SpeechClient()
```

```
audio = types.RecognitionAudio(uri=gcs_uri)
    config = types.RecognitionConfig(
        encoding=enums.RecognitionConfig.AudioEncoding.FLAC,
        sample_rate_hertz=16000,
        language_code='es-ES')


operation = client.long_running_recognize(config, audio)

    print('Waiting for operation to complete...')
    response = operation.result()  # timeout=90

    # Creamos archivo donde guardar el speech
    nombre_fichero = 'speech_' + ntpath.basename(gcs_uri) + '.txt'
    ruta = 'raw_speech/' + nombre_fichero
    f = open(ruta, 'w', encoding='utf-8')

    # Each result is for a consecutive portion of the audio. Iterate through
    # them to get the transcripts for the entire audio file.
    for result in response.results:
        # The first alternative is the most likely one for this portion.
        print(result.alternatives[0].transcript, file=f)
        print('Transcript: {}'.format(result.alternatives[0].transcript))
        print('Confidence: {}'.format(result.alternatives[0].confidence))
    f.close()
    return nombre_fichero, ruta

# transcribe_gcs('gs://tve-bucket/sonido_subido2')  # las Uris con este formato->
gs://bucket_name/object_name
```

extract_tweets.py

```
import sys
import config
import time

sys.path.append(".")


# XXX: Go to http://dev.twitter.com/apps/new to create an app and get values
# for these credentials, which you'll need to provide in place of these
# empty string values that are defined as placeholders.
# See https://dev.twitter.com/docs/auth/oauth for more information
# on Twitter's OAuth implementation.

# IMPORTANTE PONER EN CONSOLA EL COMANDO set PYTHONIOENCODING=utf-8 o bien
poner la variable de entorno directamente
def extraer_tweets(ruta='raw_tweets/'):
    import twitter

    numero_trends = config.NUMERO_TRENDS  # Indica el numero de TT's a analizar(1-20)
    numero_tweets = config.NUMERO_TWEETS  # Indica el numero de tweets por trend a obtener(min
1)
    tipo_resultado = config.TIPO_RESULTADO  # Tipo de twueets devieltos. 3 opciones: mixed recent
popular

    auth = twitter.oauth.OAuth(config.OAUTH_TOKEN, config.OAUTH_TOKEN_SECRET,
                               config.CONSUMER_KEY, config.CONSUMER_SECRET)

    fecha = time.strftime("%d_%m_%y-%H-%M", time.gmtime())
    nombre_fichero = 'tweets' + fecha + '.txt'
    ruta = ruta + nombre_fichero
    f = open(ruta, 'w', encoding='utf-8')  # Archivo donde escribir los tweets parseados

    # Initiate the connection to Twitter REST API
    twitter = twitter.Twitter(auth=auth)

    # Devuelve objeto con todos los TTs de españa
    esp_trends = twitter.trends.place(_id=23424950)

    for location in esp_trends:
        for trend in location["trends"][:numero_trends]:
            # print(trend)

            print("-#123#- %s" % trend["name"], file=f)
            print('', file=f)

            # Devuelve un diccionario de dos llaves: metadatos de la busqueda i lista de tweets en forma
de diccionarios
            search_results = twitter.search.tweets(q=trend["name"], count=numero_tweets, lang="es",
                                                   result_type=tipo_resultado,
                                                   include_entities=1, tweet_mode="extended")

            statuses = search_results['statuses']  # La chicha
            # metadata = search_results['search_metadata']

            for tweet in statuses:
                print(tweet['full_text'], file=f)

    f.close()
    return nombre_fichero, ruta
```

```
import emoji
import re
import json
import config


def limpiar(nombre, ruta):
    # Diccionario con las palabras de los tweets formato-> {tt1:{palabra:nº apariciones,...},...}
    DICCIONARIO_TWITTER = {}

    # Expresion regular para hacer match solo con los caracteres que nos interesan
    ALFABETO = re.compile('[^\W_]', re.IGNORECASE)

    # Elimina emogis de un string
    def extract_emojis(linea):
        return ''.join(c for c in linea if c not in emoji.UNICODE_EMOJI)


    archivo_origen = open(ruta, 'r', encoding='utf-8')

    lines = archivo_origen.readlines()
    lines = [line.rstrip('\n') for line in lines]
    tt_numero = 0
    for i in lines:
        if i.startswith('-#123#-'):
            tt = 'tt' + str(tt_numero)
            tt_numero += 1
            # print(tt)
            DICCIONARIO_TWITTER[tt_numero] = {}
            DICCIONARIO_TWITTER[tt_numero]['TRENDING'] = i[8:]
        lista = i.strip().split()

        for palabra in lista:
            palabra = palabra.lower()
            for ch in ['&', '#', '!', '?', '.', ',', '(', ')']:
                if ch in palabra:
                    palabra = palabra.replace(ch, '')

            if len(palabra) == len(ALFABETO.findall(palabra)) > 3 and palabra not in config.FILTRO:
                if palabra in DICCIONARIO_TWITTER[tt_numero]:
                    DICCIONARIO_TWITTER[tt_numero][palabra] += 1
                else:
                    DICCIONARIO_TWITTER[tt_numero][palabra] = 1

    archivo_origen.close()
    nombre = nombre[:-4]
    jroute = 'json_tweets/' + nombre + '.json'
    with open(jroute, 'w', encoding='utf-8') as fp:
        json.dump(DICCIONARIO_TWITTER, fp, indent=4)
    return jroute
```

limpieza_tweets.py

limpieza_speech.py

```
import re
import json
import config


def limpiar_spech(nombre, ruta):
    DICCIONARIO_SPEECH = {}

    # Expresion regular para hacer match solo con los caracteres que nos interesan
    ALFABETO = re.compile('[^\W_]', re.IGNORECASE)

    archivo_origen = open(ruta, 'r', encoding='utf-8')

    lines = archivo_origen.readlines()
    lines = [line.rstrip('\n') for line in lines]
    for i in lines:
        lista = i.strip().split()
        for palabra in lista:
            palabra_orig = palabra
            palabra = palabra.lower()
            if len(palabra) == len(ALFABETO.findall(palabra)) and len(palabra) > 3 and palabra not in
config.FILTRO:
                if (palabra in DICCIONARIO_SPEECH):
                    DICCIONARIO_SPEECH[palabra] += 1
                else:
                    if palabra_orig[0].isupper():
                        DICCIONARIO_SPEECH[palabra] = 4
                    else:
                        DICCIONARIO_SPEECH[palabra] = 1

    archivo_origen.close()

    nombre = nombre[:-4]
    jroute = 'json_speech/' + nombre + '.json'
    with open(jroute, 'w', encoding='utf-8') as fp:
        json.dump(DICCIONARIO_SPEECH, fp, indent=4)
    return jroute

# limpiar_spech('speech_Telediario-21horas-25_08_18.txt','raw_speech/speech_Telediario-21horas-
25_08_18.txt')
```

video_to_sound.py

```
import os
import ntpath


#EL PROGRAMA PUEDE TENER INPUT DE UNA URL

def extraer_sonido(ruta):
    try:
        ruta = "" + ruta + ""
        nombre_archivo = ntpath.basename(ruta)[:-4] + 'flac'
        os.system(
            "ffmpeg -y -i " + ruta + " -ar 16000 -ss 00:00:05 -t 01:00:00.0 -ac 1 -q:a 0 -map a sonido/" +
nombre_archivo)
        return nombre_archivo

    except Exception as e:
        print('Error en la conversion a sonido del archivo de video: '+e)



#extraer_sonido(ruta)
```

analisis.py

```
import json
from config import PALABRAS_GRAFICO as PG
import numpy as np
import matplotlib.pyplot as plt


def cargar(tweets='json_tweets/tweets27_08_18-03-58.json',
           speech='json_speech/speech_Telediario-21horas-25_08_18.json'):
    with open(tweets, 'r', encoding='utf-8') as fp:
        DIC_TWETS = json.load(fp)
    with open(speech, 'r', encoding='utf-8') as fp:
        DIC_SPEECH = json.load(fp)
    return DIC_TWETS, DIC_SPEECH


# Elimina el elemento TRENDING de los diccionarios de tweets
def extract_trends(dic_tweets):
    lista_trends = []
    for trend in dic_tweets:
        lista_trends.append(dic_tweets[trend]['TRENDING'])
        del dic_tweets[trend]['TRENDING']
    return lista_trends


def genera_tabla(trend, dic_tweets, dic_speech):
    tweet_list = sorted(dic_tweets.items(), key=lambda x: x[1], reverse=True)

    tw_palabras, tw_ocurrencias = map(list, zip(*tweet_list[:PG]))
    sp_ocurrencias = []
    for palabra in tw_palabras:
        if palabra in dic_speech:
            sp_ocurrencias.append(dic_speech[palabra])
        else:
            sp_ocurrencias.append(0)
    datos = [tw_ocurrencias, sp_ocurrencias]
    # print(datos)
    plt.clf()
    X = np.arange(PG)
    plt.title(trend, bbox={"facecolor": "0.8", "pad": 5})
    plt.xlabel('Palabras')
    plt.ylabel('Ocurrencias')
    p1 = plt.bar(X + 0.00, datos[0], color="b", width=0.25)
    p2 = plt.bar(X + 0.25, datos[1], color="g", width=0.25)
    plt.xticks(X + 0.38, tw_palabras, rotation=30)
    plt.legend((p1, p2), ('Twitter', 'Telediario'))
    plt.show()

    return
```

main.py

```
w = tk.Label(aut, text="Realizando speech to text y obteniendo ficheros")
        w.grid(row=4, column=0, padx=0, pady=10)
        aut.update_idletasks()
        aut.update()
        url, original_uri = subir_sonido('sonido/' + archivo_flac)
        nombre_texto, ruta_texto = transcribe_gcs(original_uri)
        jroute_speech = limpiar_spech(nombre_texto, ruta_texto)

    except:
        w = tk.Label(aut, text="Error en la conversión audio a texto del telediario", height=5, fg="red")
        w.grid(row=4, column=1, padx=0, pady=10)
        aut.update_idletasks()
        aut.update()
        aut.mainloop()
    try:
        w = tk.Label(aut, text="Generando análisis")
        w.grid(row=5, column=1, padx=0, pady=10)
        aut.update_idletasks()
        aut.update()
        dic_tweets, dic_speech = cargar(jroute_tweets, jroute_speech)
        trends = extract_trends(dic_tweets)
        aup = tk.Tk(className=' Resultados Análisis')
        aup.geometry('400x800')
        aup.iconbitmap('res/icon2.ico')
        aup.configure(background='#a1dbcd')
        for i in range(len(trends)):
            w = tk.Label(aup, text=trends[i])
            w.grid(row=i, column=0, padx=0, pady=10)
            b = tk.Button(aup, text="Resultado",
                          command=lambda i=i: genera_tabla(trends[i], dic_tweets[str(i + 1)], dic_speech))
            b.grid(row=i, column=2, padx=0, pady=10)
            aup.update_idletasks()
            aup.update()
        aup.mainloop()

    except:
        w = tk.Label(aut, text="Error en la generación de los archivos de analisis", height=5, fg="red")
        w.grid(row=2, column=1, padx=0, pady=10, columnspan=2)
        aut.update_idletasks()
        aut.update()
        aut.mainloop()

    return


def descarga_video():  # Busca la retransmision mas reciente de RTVE
    sec = tk.Tk(className=' Descarga')
    sec.geometry('400x300')
    sec.iconbitmap('res/icon2.ico')
    sec.configure(background='#a1dbcd')
    w = tk.Label(sec, text="Buscando ultima retransmision disponible del telediario de RTVE")
    w.grid(row=0, column=1, padx=10, pady=10, columnspan=2)
    sec.update_idletasks()
    sec.update()

    link, fecha = comprobar()
```

```
if (link != 0):
        def si_butt():
            w = tk.Label(sec, text="Descargando en /videos...(esto puede durar un rato)")
            w.grid(row=4, column=1, padx=0, pady=10, columnspan=2)
            sec.update_idletasks()
            sec.update()
            try:
                descarga_vid(link, fecha)
                w = tk.Label(sec, text="VÍDEO DESCARGADO", height=5)
                w.grid(row=5, column=1, padx=0, pady=10, columnspan=2)
                sec.update_idletasks()
                sec.update()
            except Exception as e:
                print(e)
                w = tk.Label(sec, text="Error en la descarga", bg='red')
                w.grid(row=5, column=1, padx=0, pady=10, columnspan=2)
                sec.update_idletasks()
                sec.update()

        w = tk.Label(sec, text="Se ha encontrado: " + fecha)
        w.grid(row=1, column=1, padx=0, pady=10, columnspan=2)
        w = tk.Label(sec, text="Descargar vídeo?")
        w.grid(row=2, column=1, padx=0, pady=10, columnspan=2)
        b2 = tk.Button(sec, text="Si", command=si_butt)
        b2.grid(row=3, column=1, padx=10, pady=10)
        b3 = tk.Button(sec, text="No", command=lambda: sec.destroy())
        b3.grid(row=3, column=2, padx=0, pady=10)
        sec.mainloop()
    else:
        w = tk.Entry(sec, text="Ha habido un problema con la busqueda")
        w.grid(row=4, column=1, padx=0, pady=10)
        time.sleep(5)
        sec.destroy()
    return


def obtener_tweets():
    try:
        twe = tk.Tk(className=' Obtener tweets')
        twe.geometry('400x300')
        twe.iconbitmap('res/icon2.ico')
        twe.configure(background='#a1dbcd')
        print('Extrayendo....')
        w = tk.Label(twe, text="Descargando tweets...")
        w.grid(row=0, column=0, padx=0, pady=10)
        twe.update_idletasks()
        twe.update()
        nombre_fichero, ruta = extraer_tweets()
        w2 = tk.Label(twe, text="Fichero de texto en: " + ruta)
        w2.grid(row=1, column=0, padx=0, pady=10)
        twe.update_idletasks()
        twe.update()
        w3 = tk.Label(twe, text="Limpiando tweets y generando json...")
        w3.grid(row=2, column=0, padx=0, pady=10)
        twe.update_idletasks()
        twe.update()
        jroute = limpiar(nombre_fichero, ruta)
        w4 = tk.Label(twe, text="Fichero json en: " + jroute)
        w4.grid(row=3, column=0, padx=0, pady=10)
        twe.mainloop()
```

```
    except Exception as e:
        print(e)
        w = tk.Label(twe, text="Error en la obtencion de los tweets", bg='red')
        w.grid(row=5, column=1, padx=0, pady=10, columnspan=2)
        twe.mainloop()
    return


def obtener_audio():
    aud = tk.Tk(className=' Obtener archivos de speech')
    aud.geometry('400x300')
    aud.iconbitmap('res/icon2.ico')
    aud.configure(background='#a1dbcd')
    w = tk.Label(aud, text="Selecciona un archivo de vídeo (formato .mp4) para extraer")
    w.grid(row=0, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    time.sleep(1)

    filename = askopenfilename(parent=aud, initialdir=ntpath.abspath("videos").replace("\\", "/"),
                               title="Selecciona el archivo MP4",
                               filetypes=(("MP4 files", "*.mp4"), ("all files", "*.*")))
    w = tk.Label(aud, text="Extrayendo audio con ffmpeg...")
    w.grid(row=1, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    archivo_flac = extraer_sonido(filename)
    w = tk.Label(aud, text="Subiendo archivo a Google Cloud para su analisis...")
    w.grid(row=2, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    url, original_uri = subir_sonido('sonido/' + archivo_flac)
    w = tk.Label(aud, text="Archivo subido, analizando...(esto puede tardar un rato)")
    w.grid(row=3, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    nombre_texto, ruta_texto = transcribe_gcs(original_uri)
    # print(archivo_flac)
    # print(nombre_texto, ruta_texto)
    w = tk.Label(aud, text="Archivo de texto en " + ruta_texto)
    w.grid(row=4, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    w = tk.Label(aud, text="Generando archivo JSON a partir del texto...")
    w.grid(row=5, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    ruta_json = limpiar_spech(nombre_texto, ruta_texto)
    w = tk.Label(aud, text="Archivo generado en " + ruta_json)
    w.grid(row=6, column=0, padx=0, pady=10)
    aud.update_idletasks()
    aud.update()
    return


def analisis_personalizado():
    aup = tk.Tk(className=' Análisis personalizado')
    aup.geometry('400x300')
    aup.iconbitmap('res/icon2.ico')
    aup.configure(background='#a1dbcd')
    w = tk.Label(aup, text="Selecciona el fichero JSON con los tweets")
    w.grid(row=0, column=0, padx=0, pady=10)
    aup.update_idletasks()
    aup.update()
```

```
    json_tweets = askopenfilename(parent=aup, initialdir=ntpath.abspath("json_tweets").replace("\\",
"/"),
                                  title="Selecciona el archivo JSON",
                                  filetypes=(("JSON files", "*.json"), ("All files", "*.*")))
    w = tk.Label(aup, text="Selecciona el fichero JSON perteneciente al telediario")
    w.grid(row=0, column=0, padx=0, pady=10)
    aup.update_idletasks()
    aup.update()
    time.sleep(1)
    json_speech = askopenfilename(parent=aup, initialdir=ntpath.abspath("json_speech").replace("\\",
"/"),
                                  title="Selecciona el archivo JSON",
                                  filetypes=(("JSON files", "*.json"), ("All files", "*.*")))
    dic_tweets, dic_speech = cargar(json_tweets, json_speech)
    trends = extract_trends(dic_tweets)
    aup.destroy()
    aup = tk.Tk(className=' Resultados Análisis')
    aup.geometry('400x800')
    aup.iconbitmap('res/icon2.ico')
    aup.configure(background='#a1dbcd')
    for i in range(len(trends)):
        w = tk.Label(aup, text=trends[i])
        w.grid(row=i, column=0, padx=0, pady=10)
        b = tk.Button(aup, text="Resultado",
                      command=lambda i=i: genera_tabla(trends[i], dic_tweets[str(i + 1)], dic_speech))
        b.grid(row=i, column=2, padx=10, pady=10)
        aup.update_idletasks()
        aup.update()
    aup.mainloop()
    return

def exit():
    tk._exit(0)

def main():
    top = tk.Tk(className=' Herramienta de Análisis de Telediarios')
    top.geometry('450x300')
    top.iconbitmap('res/icon2.ico')
    top.configure(background='#a1dbcd')
    b1 = tk.Button(top, text="Análisis Automático", command=analisis_auto, fg='#383a39',
bg="#21dbc1")
    b1.grid(row=0, column=1, padx=160, pady=10)
    b2 = tk.Button(top, text="Descargar Vídeo", command=descarga_video)
    b2.grid(row=1, column=1, pady=10)
    b3 = tk.Button(top, text="Obtener Tweets", command=obtener_tweets)
    b3.grid(row=2, column=1, pady=10)
    b4 = tk.Button(top, text="Obtener Audio", command=obtener_audio)
    b4.grid(row=3, column=1, pady=10)
    b5 = tk.Button(top, text="Análisis Personalizado", command=analisis_personalizado)
    b5.grid(row=4, column=1, pady=10)
    b6 = tk.Button(top, text="Salir", command=exit)
    b6.grid(row=5, column=1, pady=10)
    top.mainloop()


if __name__ == "__main__":
    main()
```

Anexo II. Ejemplo de uso

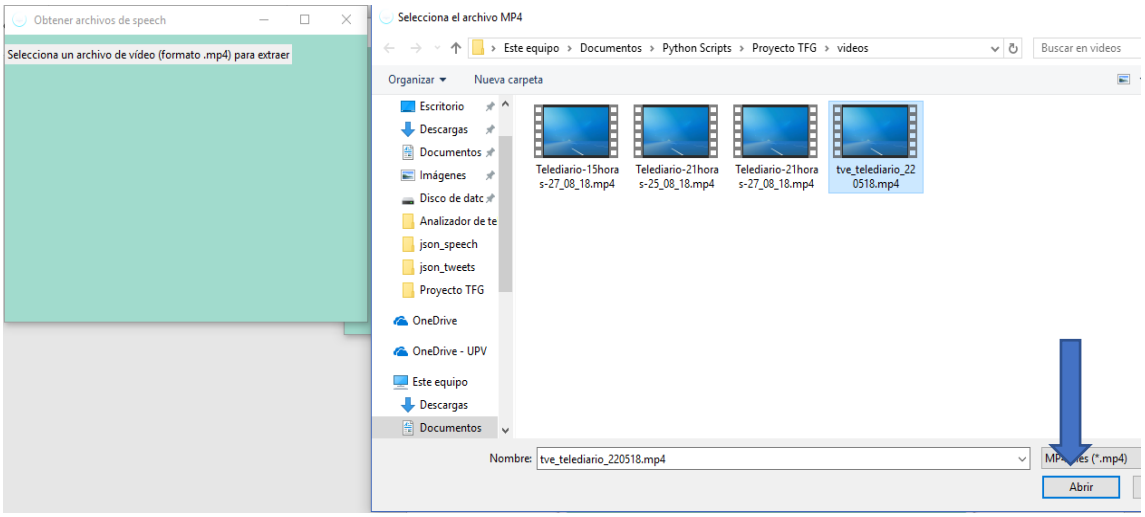
Este ejemplo va a ilustrar como realizar un análisis personalizado de las redes sociales, si lo que se desea realizar es un análisis automático solamente hay que pulsar el primer botón de la ventana principal de la aplicación.

Con todas las dependencias y librerías instaladas tal y como se explica en la guía, situarse en la carpeta donde se encuentran los archivos del programa y ejecutar el comando:

```
python main.py
```

Aparecerá la ventana principal, para obtener el ultimo telediario disponible pulsar el botón ‘Descargar Vídeo’ y confirmar el archivo a descargar.

Una vez descargado procedemos a transcribir el audio, para ello pulsamos el botón ‘Obtener Audio’ y seleccionamos el archivo de video a transcribir como se nos indica en la pantalla de mensajes.

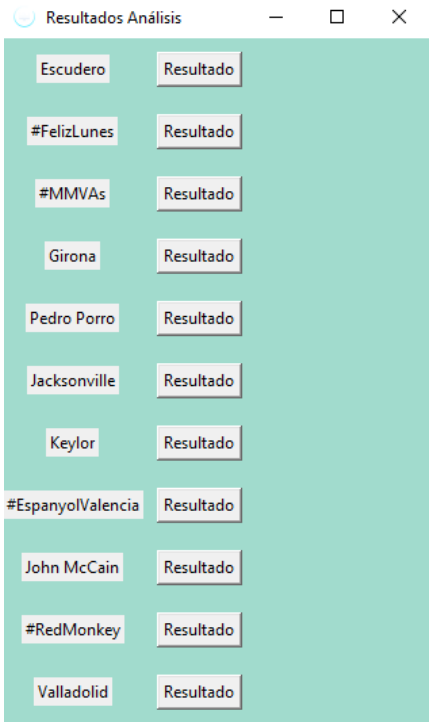


Confirmamos con el botón abrir y esperamos a que se complete la operación que generará el archivo necesario para el análisis.

Seguidamente obtenemos los tópicos de las redes sociales con el botón ‘Obtener tweets’. Esta operación es automática y generará el otro archivo necesario para el análisis.

El orden de estas dos últimas operaciones es indiferente.

Finalmente presionamos el botón de ‘Análisis Personalizado’ y seleccionamos los archivos que la herramienta acaba de crear de las ventanas de selección. Aparecerá entonces la ventana que nos permite ver los resultados.



Anexo III. Herramientas de la plataforma de análisis de telediarios.

Análisis estadístico y de sentimiento de las palabras empleadas en los informativos

Autor: José Joaquín Rodríguez García

Consistirá en contar las palabras usadas en referencia a cada partido o referente político. De este conteo se establecerán conexiones entre grupos de palabras usando como medio de conexión si son sinónimas o antónimas entre ellas.

También se van a emplear técnicas de análisis de sentimientos usando diccionarios de afectos SDAL para averiguar si el comunicador mantiene una opinión neutra respecto a los diferentes partidos políticos sin intentar sugestionar al presentador determinadas ideas. Esto se va a realizar mediante contrastes de hipótesis estadísticas para verificar si existen diferencias estadísticamente significativas en la presentación de la información.

Estos dos análisis presentaran sus datos en forma de gráficos y tablas dentro de dicha plataforma

Análisis de la tipología de las imágenes mostradas en el informativo

Autora: Araceli Teruel Doménech

Este proyecto se basa en la idea de que mediante diferentes técnicas de imagen y cinematografía se puede alterar la percepción que se tiene de una persona ya sea mejorando o empeorando la opinión que tiene de ella el telespectador

La función de esta herramienta será estudiar las imágenes que ofrecen los telediarios para analizar el impacto que tienen sobre las personas, los posibles datos que se puedan sacar de ellas. Los datos que vamos a estudiar principalmente serán, los planos, los ángulos, las personas que salen, el tiempo que aparecen y el orden en el que aparecen.

Este análisis se va a realizar mediante técnicas de inteligencia artificial basadas en el reconocimiento de imagen usando modelos predictivos y clasificadores estadísticos. Se van a extraer las imágenes de la retransmisión del informativo donde, principalmente, aparezcan líderes políticos y se va a averiguar qué diferencias existen potencialmente en la representación que se da de ellos al público basándose en los datos a estudiar anteriormente mencionados.

Estos resultados se van a presentar como una estadística respecto a cada líder político en forma de gráficos.

Análisis del mensaje transmitido por representantes políticos

Autor: Ricardo Miguel Cancar de Abreu

Este proyecto busca esclarecer si el mensaje transmitido por los líderes políticos en los telediarios concuerda con los puntos que se definen en el programa electoral de su partido o si por lo contrario contradicen su mensaje con las promesas que realizaron a sus votantes.

El desarrollo se basa en un sistema de análisis de contenido que permita clasificar algunas de las promesas de los principales actores políticos y compararlas con lo dicho en el congreso de diputados. La idea es usar las técnicas de *big data* para procesar el audio de los telediarios y emplear herramientas de reconocimiento de voz para distinguir la voz de algunos de los políticos más importantes de España, una vez identificada la voz de un político transcribir lo que transmita dicho político de audio a texto y así de poder identificar mediante el uso de expresiones regulares, si el político menciona algún tema relacionado con su plan de gobierno en los telediarios.

Se planea comparar lo dicho por el político con la información registrada por el en la página dedicada a almacenar todos los temas tratados en el congreso de diputados y realizar análisis estadísticos sobre dichos temas.

Los resultados se van a presentar en forma de representación estadística de los puntos del programa electoral de cada partido contra la confianza de que el líder político de tal partido haya dicho algo que la apoye.