Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Búsqueda semántica de perfiles en redes sociales

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Diego Ros Pagán

Tutor: José Ángel González Barba

Tutor: Ferran Pla Santamaría

Curso 2020-2021

Resumen

En este proyecto se ha desarrollado una aplicación que permite identificar la temática predominante en usuarios de la red social *Twitter* mediante el análisis del contenido textual generado por el usuario atendiendo principalmente a las relaciones semánticas que presentan los textos. Se utiliza un algoritmo de clasificación basado en *word embeddings*, en concreto se usa un modelo preentrenado *Word2Vec* para calcular la distancia semántica entre los temas, cuentas y *tweets*.

El proyecto se centra en la red social *Twitter*, utilizando los *tweets* como la unidad básica a clasificar. Mediante un conjunto semilla de cuentas etiquetadas manualmente por temática se crea un modelo de representación vectorial. El modelo está creado en función de las representaciones de las cuentas en las que predominan dichos temas. Este modelado nos ha permitido crear una aplicación para clasificar cuentas de usuarios y *tweets* de forma automática en los temas considerados.

Palabras clave: word embedding, Word2Vec, Twitter, clasificación semántica.

Abstract

The objective of this project is to develop an application which will allow to identify the predominant topic in user profiles on *Twitter* social network. This is done by analysing the textual content generated by the users, mainly paying attention to the semantic relationships presented in the texts. A classification algorithm based on *word embeddings* is used. Concretely, a pre trained *Word2Vec* model is used to calculate the semantic distance between the topics, accounts and *tweets*.

The project is focused within the *Twitter* social network, using *tweets* as the basic unit to be classified. By a seed set of accounts manually labelled by topic, a vector representation model is created. The model is created based on the representations of the accounts in which these topics predominate. This modelling allows us to create an application to classify user accounts and *tweets* automatically into the topics considered.

Keywords: word embedding, Word2Vec, Twitter, semantic classification.

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

La clasificación de textos no es un tema reciente, ya que la propia existencia de la escritura implica una necesidad de clasificar la misma, ya sea por autor, temática idioma u otros criterios siendo siempre la finalidad de esta filtrar los textos por características importantes. La clasificación de textos puede realizarse según diferentes ámbitos o razones formales. Vamos a presentar varias de estas:

- Tipo de canal: oral o escrito
- Situación: formal o informal
- Intención comunicativa: informativo, persuasivo, directivos, o retóricos.
- Modo: narrativo, descriptivo, dialogado, expositivo o argumentativo
- Ámbito de uso: literario, científico-técnicos, humanísticos, jurídico-administrativos, periodísticos o publicitarios.
- Temática

En la antigüedad los textos usualmente se clasificaban en archivos o bibliotecas, predominando una clasificación sobre los temas correspondientes a la actividad religiosa, política, o económica y administrativa. En la actualidad, el gran volumen de información que nos proporciona internet implica que no se puede clasificar esta de forma manual, Por ejemplo, en la Wikipedia en inglés hay más de seis millones de artículos. Además, ha surgido la necesidad de clasificar textos en línea, como puede ser saber si un email recibido es *spam* o no.

Las técnicas de procesamiento del lenguaje natural (PLN), entre otras cosas, pueden ser usadas para solucionar estos problemas. Uno de los enfoques usados tradicionalmente en la clasificación de textos mediante técnicas de PLN es *Bag of Words* (BoW) o bolsa de palabras, que consiste en la indexación y cálculo de la frecuencia de los términos de un documento. En cambio, en este proyecto se van a usar técnicas enfocadas en clasificación semántica distribucional (*word embeddings*). Estas técnicas tratan de reproducir las relaciones semánticas entre las palabras mediante su transformación en vectores calculados respecto al contexto en el que se encuentran.

En el proyecto aplicamos estas técnicas en textos de redes sociales, clasificando cuentas de usuarios en redes sociales según los textos que han escrito. La clasificación que se ha estudiado es según la temática, es decir, basado en los temas principales que tratan las cuentas.

De esta forma el principal objetivo del proyecto consiste en realizar una aplicación que, mediante herramientas de procesamiento del lenguaje natural, permita analizar la temática de diferentes cuentas de *Twitter*, además de presentar una visualización de los resultados de forma sencilla y amigable para el usuario.

En este capítulo se aborda brevemente en qué consiste el proyecto realizado y a que pretende responder, tras lo que se comentan las motivaciones personales que me han llevado a realizarlo. A continuación, se describen los objetivos principales y secundarios del proyecto. Finalmente se describe la estructura del documento.

1.1. Motivación

La motivación para realizar este proyecto la podemos centrar en tres aspectos: sociales, laborales y de aprendizaje, que han sido de especial relevancia para mí en el transcurso del proyecto.

Con respecto al aspecto social, las redes sociales son importantes para un gran número de personas en la actualidad. La información que recibimos de estas afecta a nuestra percepción de la vida y las decisiones que tomamos. Personalmente creo que es importante poder filtrar toda la información que recibimos de las redes sociales, permitiéndonos acceder a la información relevante para cada uno.

Con respecto al aspecto laboral, el procesamiento del lenguaje natural es un campo de la informática que es de gran importancia en la actualidad, con muchas aplicaciones diferentes e interesantes. Considero que este proyecto me va a ayudar a acercarme a las diferentes salidas laborales en las que estoy interesado, proporcionándome una mejor preparación para afrontar con éxito mis objetivos.

Con respecto al aspecto de aprendizaje, a lo largo de la carrera he estudiado diferentes asignaturas, las cuales me han dado formación necesaria para el desempeño de la informática. Mediante este proyecto creo que voy a poder profundizar más en la rama de computación, permitiéndome aprender técnicas que no conocía o no había utilizado previamente, y poner en práctica las técnicas, algoritmos, buenas prácticas y otros conocimientos que he aprendido a lo largo de la carrera.

1.2. Objetivos

Los objetivos fijados en este proyecto son los siguientes:

- Desarrollar una aplicación que permita clasificar diferentes cuentas de *Twitter*, considerando la semántica de las palabras que componen los *tweets*, y permitiendo visualizar los resultados de forma amigable y clara a los usuarios.
- Recuperación de *tweets* de forma automática, permitiendo crear una base de datos con los *tweets,* clasificados según su temática, que se van a usar en la aplicación.
- Crear un modelo que permita clasificar cuentas de *Twitter* en diferentes temáticas, evaluando el uso de diferentes técnicas para obtener un mejor clasificador.
- Crear una aplicación sencilla y amigable para el usuario, separando en capas la lógica de la aplicación y la interfaz de usuario.
- Informar al usuario de qué cuentas de *Twitter* y *tweets* son más similares a la búsqueda que se realice.

1.3. Estructura

A continuación, se describe la estructura de la memoria del proyecto indicando el contenido de cada uno de los capítulos.

- **1 Introducción:** En este capítulo se introduce el trabajo realizado, también se describe las motivaciones personales que me llevaron a plantear y

desarrollar el proyecto. Además, se marcan los objetivos del proyecto y se describe la estructura de la memoria.

- **2 Estado del arte:** En este capítulo se describe la problemática de la clasificación de textos, su historia y su estado actual, centrándonos en la automatización mediante el procesamiento del lenguaje natural. Además, se discuten técnicas de detección de tópicos y perfilado de autor que pueden servir para la clasificación de textos.

- **3 Preproceso y representación de texto :** Este capítulo presenta la problemática del procesamiento del lenguaje natural centrándonos en las técnicas más utilizadas para la clasificación de textos. Primero se presenta el preprocesado de textos, así como, los principales métodos de representación de palabras y frases: one-hot, *bag of words* y *word embeddings,* centrándonos en el último, pues es el usado en el proyecto.

- **4 Identificación del problema y propuesta de solución:** Este capítulo introduce el problema al que se trata de dar solución mediante la aplicación desarrollada. Además, se explica en qué consiste la aplicación propuesta y cómo se estructura su arquitectura y las dependencias entre las distintas partes de la aplicación. Finalmente, se explican las diferentes tecnologías usadas durante su realización.

- **5 Desarrollo de la solución:** Es el capítulo más extenso de proyecto, pues explica cómo se ha desarrollado la aplicación propuesta de forma exhaustiva, qué pasos se han seguido y qué pruebas y modificaciones se han llevado a cabo para validar su correcto funcionamiento. Además de presentar los resultados, gráficos de la misma.

- **6 Conclusiones:** En este capítulo se detallan las conclusiones y resultados del proyecto. Además, se explica qué asignaturas cursadas a lo largo del grado en Ingeniería Informática han sido de especial importancia para poder desarrollar la aplicación.

- **7 Trabajos futuros:** En este capítulo se proponen varias opciones de trabajos académicos futuros que se podrían realizar para mejorar o complementar a este.

- **Glosario:** En esta sección se describen términos específicos del proyecto, que permiten una lectura más rápida y sencilla a personas no familiarizadas en este ámbito.

- **Apéndice 1: Cuentas de *Twitter*.** Apéndice que incluye las tablas con las cuentas de *Twitter* usadas para la creación del modelo de clasificación y su testeo, indicando los *tweets* recuperados de cada una.

2. Estado del arte

En este capítulo se introducen los enfoques clásicos para la clasificación de textos, sus metodologías y formatos. A continuación, trataremos la clasificación de textos en la actualidad, y cómo el uso de técnicas de procesamiento del lenguaje natural ha permitido automatizar la clasificación, explicando diferentes técnicas y planteamientos. Finalmente, presentamos algunos trabajos que usan la detección de tópicos y el perfilado de autores, centrándonos en las similitudes de estos con la clasificación de textos.

2.1. Clasificación de textos en la historia: Bibliotecas

El proceso de clasificación de textos es un proceso paralelo a la existencia de la escritura, los libros y las bibliotecas. Con la creación de la escritura nació la necesidad de clasificar los textos para poder filtrar textos según los criterios deseados.

Hay varias culturas antiguas que hicieron uso de las bibliotecas para clasificar y almacenar textos. Las más antiguas que existieron y de las que se tiene conocimientos son las siguientes:

- En las ciudades mesopotámicas nacieron archivos en los templos, estos tenían una función conservadora, de registro de hechos ligados a la actividad religiosa, política, económica y administrativa.
- En el antiguo Egipto existieron dos clases de instituciones que clasificaban textos. Las Casas de los Libros, que archivaban la documentación administrativa, y las Casas de la Vida, que eran consideradas centros de estudios y hacían copias y colecciones en las diferentes formas de escritura: jeroglífica, hierática o demótica.

En 1876 un bibliotecario llamado Melvil Dewey creo un sistema decimal para la clasificación de libros, llamado en la actualidad “Sistema de Clasificación Decimal Dewey”. Definió diferentes categorías o tópicos en los que se puede categorizar los textos, y les asignó un número de tres dígitos, como se puede observar en la *ilustración 1*.

Ilustración 1: Sistema de clasificación decimal Dewey

Este sistema tiene muchas y diversas subcategorías dentro de cada categoría general. Esta clasificación sigue siendo muy utilizada en las bibliotecas en la actualidad.

2.2. Clasificación automática de textos.

El gran volumen de información en internet dificulta o incluso imposibilita la clasificación de textos manual. Por ello, cada vez más se están usando técnicas de Procesamiento del Lenguaje Natural (PLN) para la clasificación de textos. Estas técnicas surgieron en la década de los 60, pero en los últimos años ha habido un aumento substancial de su estudio y uso, por lo que este apartado se va a centrar en estudios recientes.

La clasificación de textos mediante técnicas de PLN tiene muchas aproximaciones y usos diferentes que, se han ido desarrollando de forma paralela. Dentro de la clasificación de textos, vamos a centrarnos en las aproximaciones más relacionadas con el proyecto realizado:

- Clasificación de tópicos
- Perfilado de autores, identificando los temas predominantes en su comunicación a través de redes sociales.

Es conveniente notar que la clasificación de textos no se limita únicamente a estas problemáticas, se ha usado PLN para muchas otras y diversos objetivos: clasificación por sentimientos, clasificación de emails en spam, clasificación por idioma, etc.

2.2.1. Clasificación de tópicos

En los últimos años, para clasificar documentos en diferentes tópicos, se han utilizado diferentes técnicas de procesamiento del lenguaje natural y de aprendizaje automático. Tradicionalmente, se ha usado la representación *BoW* para clasificar

documentos respecto a la temática que presentan, pero esta, al ser una representación simbólica no tiene en cuenta muchas de las propiedades del lenguaje natural como: las relaciones semánticas entre las palabras, y las relaciones temporales entre estas.

Los autores de [1], [2] hicieron en 2009 y 2011 diversos estudios de clasificación de textos por temática en Wikipedia basados en *Bag of words,* pero introduciendo cambios para mejorar los resultados respecto a otros estudios previos.

Más concretamente en [1] se construye un tesauro de conceptos de Wikipedia para mantener relaciones semánticas entre las palabras: sinonimia, hiponimia, y relaciones asociativas. Los autores basaron el aprendizaje automático realizado en el formalismo de las Máquinas de Vectores Soporte. Finalmente, en el documento se demostró que la introducción de relaciones semánticas en la clasificación de textos mejora el modelo resultante.

En cambio, en [2], se estudia la clasificación *BoW* respecto a una especificación temática previa, es decir, para cada tema estudiado se calcula la frecuencia de varios términos representativos, como los mostrados en la *ilustración 2*, y a partir de operaciones sencillas e indicando un conjunto de pesos se procede a crear el modelo de clasificación. Los pesos se calculan dependiendo del tipo de palabra y la función que tiene en el conjunto de documentos: si representa a más de un tema, provoca ambigüedad o es muy representativo. En conclusión, el modelo permite ajustar la clasificación de documentos al conjunto temático introducido por el usuario concreto. En el documento se presentan resultados competitivos incluso en condiciones desfavorables, es decir, cuando las temáticas a clasificar comparten vocabulario pues parten del mismo tema general: deporte

Categoría	Términos	Palabras clave
Atletismo	Atletismo	Altura, longitud, metros, m, kilómetros, carrera, correr
Ciclismo	tour, ciclista	Tour, etapa, giro, crono, vuelta
Motor	F1, Fórmula 1	Pole, Fórmula 1
Golf	golf, golfista	Golf, PGA, green, golpes
Tenis	tenis, grand slam	Atp, tenis
Cultura	Cultura	

Ilustración 2: Uso de términos representativos para BoW en [2]

Los autores de [3] exponen el uso del modelado de tópicos en el ámbito de la bioinformática, basados en algoritmos BoW y técnicas *Latent Dirichlet Allocation* (LDA), que permiten inferir probabilísticamente los temas de los documentos. En [3], de forma similar a [2] se definen palabras clave que representan un tema, y dependiendo de la frecuencia de las palabras clave se entrena un clasificador basado en algoritmos LDA. En [4] se presenta el uso del algoritmo LDA y *BoW*, de manera similar a [3], pero aplicado a redes sociales, Este trabajo concluye que el sistema desarrollado es un método sencillo, eficaz y rápido para la clasificación automática de textos en redes sociales e indicando que esta aun presenta margen de mejora si se especifica un conjunto de *stopwords* relativo al conjunto de temas a clasificar.

En [5] se estudia el uso del algoritmo de *Naive Bayes* junto a técnicas de PLN para clasificar diversos artículos periodísticos. Los autores estudian el uso de varias técnicas de preproceso de texto, como: *stemming*, eliminación de *stopwords* y reconocimiento de nombres. Adicionalmente, experimentan con sistemas basados en máxima entropía.

Con esta aproximación obtiene un rendimiento muy prometedor en la clasificación usando siete tópicos.

Además de clasificar un documento en los diversos tópicos que trata, otra aproximación consiste en descubrir automáticamente los temas que trata un documento. A continuación, se van a presentar dos trabajos que hacen uso de estas técnicas.

En [6] se realiza un estudio de las técnicas usadas para *Topic Detection and Tracking* (TDT) y se proponen algunos cambios y mejoras respecto a estudios realizados previamente. El problema TDT se puede dividir en tres tareas: segmentar un texto en diferentes historias más pequeñas, identificar los eventos que suceden en estas historias, y dado un número de historias sobre un evento identificar las siguientes historias que continúan la tendencia. Es decir, el TDT identifica tópicos o eventos (p.ej. la segunda guerra mundial) en historias, a partir de un número reducido de textos que comparten el tópico.

Los autores de [6] exponen tres algoritmos distintos de TDT y su funcionamiento en las diferentes tareas, incidiendo en su uso para la detección de eventos en línea, lo que implica capturar nuevos tópicos o eventos no predefinidos a partir del texto de entrada.

En [7] se realiza un estudio exhaustivo de detección de tópicos y análisis de sentimientos en *tweets* en español. Los autores comparan varias técnicas y combinaciones de preproceso de texto, estudio de información específica de *Twitter* y técnicas de aprendizaje automático como: n-gramas, lematización, corrección de palabras, *Hashtags*, autores, *IBK*, *Naïve Bayes*, etc.

Configuration number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Parameters														
n-gram	1	1	1	1	1	1	1	2	1	1	1	1	2	1
Lemma/Stem (**L/S**)	L	L	S	L	L	L	L	L	L	L	L	L	L	L
SMS					X						X		X	
Word types (Nouns C&P)	X		X	X	X	X	X	X	X	X			X	X
Correct words				X										
Hashtags	X	X	X	X	X		X	X	X	X	X	X		X
Author Tags	X	X	X	X	X	X		X	X		X	X	X	X
Links									X	X				
Classifiers (Accuracy)														
Ibk	36.62	30.54	36.37	36.62	36.77	31.17	**37.97**	32.64	**38.57**	32.47	30.49	30.54	33.83	36.62
ComplementNaiveBayes	56.75	**58.45**	56.25	56.75	**57**	55.75	53.66	53.56	53.56	51.67	**58.25**	**58.45**	52.02	56.75
NaiveBayesMultinomial	56.35	**57.1**	55.61	56.35	56.25	55.46	53.71	55.61	54.11	53.26	**56.95**	**57.1**	56	56.35
RandomCommittee	53.56	52.47	52.62	53.56	**53.91**	**53.66**	52.52	**55.06**	52.72	52.27	51.92	52.47	38.15	53.56
SMO	56.3	55.06	55.95	56.3	56.55	55.51	55.26	55.9	55.16	54.21	42.38	55.06	54.81	56.3

Ilustración 3: Accuracy (%) de detección de tópicos con diferentes configuraciones en [7]

Estas técnicas están basadas en estudios previos realizados en datos en inglés, con resultados bastante positivos. Los autores tratan de extrapolarlas a su uso en español, pues al ser idiomas sintácticamente muy diferentes estas técnicas no tienen por qué ser las más adecuadas. Finalmente concluyen que ninguna de estas técnicas produce una mejora significativa en los resultados. Como se puede ver en *ilustración 3* el resultado con mayor *accuracy* es 58%, por lo que los autores creen que es necesario más estudio en el campo, pues estos resultados muestran que hay mucho margen de mejora.

2.2.2. Perfilado de autores

Para clasificar un texto puede ser importante conocer las características principales de su autor: su edad, sexo, intereses, etc. En redes sociales esto es aún más importante, pues hay ocasiones en las que clasificar un autor es equivalente a clasificar un conjunto

de temas o intereses, principalmente si se está hablando de revistas, o grupos centrados en ciertos hobbies o campos de estudio.

En [8], se estudian diversas características de los autores en *Twitter*, principalmente su género y edad, mediante Máquinas de Vectores Soporte y representación *BoW*. Se centra en *tweets de diversos idiomas:* inglés, español, italiano y alemán, El principal objetivo es usar un lexicón que enriquezca las representaciones de las palabras en el dominio de la red social *Twitter*.

Los autores de [9] presentan un estudio detallado sobre el perfilado de autores en Facebook, usando textos en portugués brasileño. Los autores estudian varias características de los usuarios, incluyendo su género y edad, junto con su conocimiento sobre las Tecnologías de la Información y su interés religioso. Es interesante resaltar que existe una estrecha relación entre los dos últimos aspectos y la temática predominante en sus publicaciones en Facebook. Además, realizan un comparativa respecto a diferentes técnicas para su cálculo: *BoW*, *Word2Vec*, *TD-IDF*, etc. En los resultados de la experimentación se observa cómo TF-IDF proporciona un mejor valor F1 y siendo *Word2Vec* el segundo mejor.

En [10] se realiza un estudio sobre de técnicas de perfilado de autor, engaño y detección de ironía en diferentes dialectos de árabe. El objetivo principal es estudiar técnicas para la ciberseguridad y detección de cuentas de usuarios en redes sociales posiblemente terroristas. En este estudio se comparan diferentes técnicas de los temas indicados y comentando para cada uno qué técnica o conjunto de técnicas mejora la clasificación realizada.

En [11] los autores realizan un estudio sobre la creación de ideas sobre el suicidio en redes sociales mediante técnicas de perfilado de autores y grafos de redes sociales. En este estudio los autores realizan tres aportaciones importantes a este campo: primero crean un conjunto de datos clasificado manualmente sobre patrones del comportamiento suicida junto con un histórico de *tweets* y un grafo de redes sociales, segundo proponen la arquitectura SNAP-BATNET, la cual se basa en apilar características hechas a mano sobre: perfilado de autores, grafos de *embeddings* de redes sociales, características de datos históricos y metadatos de los *tweets.* Su tercera aportación es realizar una comparativa extensa sobre algoritmos y representaciones tradicionales, como: TF-IDF, POS, GLoVe *embeddings,* NCR y LDA. En los resultados de la experimentación los autores concluyen que la arquitectura propuesta mejora particularmente los resultados de la medida del *recall*.

3. Preproceso y representación de texto

El procesamiento del lenguaje natural (PLN) es la disciplina que usa la computación para el análisis del lenguaje. Esto se consigue usando diferentes técnicas para el análisis y representación de texto a diferentes niveles. El objetivo final es el procesado del lenguaje de forma similar a la que hacemos los humanos. Dicho procesamiento puede ser útil en distintas aplicaciones: recuperación de información, clasificación de textos, traducción automática, etc.

En este apartado se presentan las técnicas de preprocesamiento que permiten abordar los problemas de manera más sencilla y eficiente. Después, se describirán varios métodos de representación de texto, que van desde las representaciones discretas tradicionales *one-hot* y bolsa de palabras, hasta las representaciones continuas de palabras (*word embeddings)* que permiten modelar relaciones "semánticas". Estas últimas representaciones son más utilizadas en la actualidad debido a su buen comportamiento en problemas de PLN y son las utilizadas en el proyecto.

3.1. Técnicas de preprocesamiento de texto

Usualmente las soluciones de PLN realizan ciertas tareas de preproceso que permiten normalizar el texto para así obtener mejores modelos de representación. A continuación, se van a explicar algunas de las técnicas de preprocesado más utilizadas.

- **Tokenizacion:** un *token* es una unidad de significado del texto, usualmente *tokenizar* hace referencia a separar un texto según sus frases, palabras, o unidades de significado. P.ej. Nueva York podría considerarse un único token. Algunas tareas de PLN hacen uso de la agrupación de *tokens*, esto es conocido como *n-gramas*.
- **Eliminación de stopwords:** las *stopwords* o palabras vacías, son palabras de uso abundante en un idioma que no aportan significado a las oraciones, por lo que pueden modificar o sesgar excesivamente los modelos de PLN. Las *stopwords* se suelen eliminar dependiendo del modelo deseado, para evitar el sesgo o efectos no deseados que provocan. Usualmente se consideran como *stopwords* los artículos, pronombres y preposiciones, pero no hay una lista exacta de ellas.
- **Stemming:** consiste en reducir las palabras a su raíz. P.ej. bibliotecario y biblioteca se reduciría a '*bibliotec'.* Se suele usar cuando las variaciones sintácticas no aportan significado al modelo que se desea obtener. El uso de *stemming* en sistemas de recuperación de información disminuye considerablemente los ficheros de índices y puede aumentar las prestaciones del sistema de recuperación.
- **Lematización:** es un proceso que consiste en obtener el lema de las palabras con el fin de reducir la talla del vocabulario. Por ejemplo, el lema de los sustantivos se obtiene cambiando su número a singular, en adjetivos se obtienen en singular masculino, y en verbos se utiliza como lema el tiempo verbal infinitivo. La lematización requiere realizar un análisis morfológico de las palabras. P. ej. canto, cantas, y canta son formas verbales del verbo cantar y serían lematizados a cantar; niña, niño, niñita, niños, y niñotes serían lematizados a niño.

3.2. One-hot

"El concepto de representación *one-hot* proviene de la teoría de circuitos digitales, donde intuitivamente se puede ver como un conjunto de bits de tal forma que solo uno de estos bits se encuentra activo (1) y todos los demás anulados (0)" [12]

Este consiste en representar cada palabra de forma vectorial, $\mathrm{x} = \{x_1, x_2, \ldots, \mathrm{x}_n\}$, siendo n la talla del vocabulario, de manera que, para cada palabra del vocabulario, la representación del componente $x_i = 1$ si, siendo i y j las palabras del vocabulario y frase respectivamente, $i = j$ en caso contrario es 0.

En la *tabla 1* se muestra la representación *one-hot* para la frase "El rey se puso el anillo", donde se puede observar una matriz resultante muy dispersa. Cada fila de la matriz es la representación *one-hot* vectorial de cada token, por ejemplo, en el caso de la palabra "anillo" el vector resultante sería: $x_{anillo} = \{0,0,0,0,0,1\}$.

	El	**rey**	**se**	**puso**	**el**	**anillo**
El	1	0	0	0	1	0
REY	0	1	0	0	0	0
SE	0	0	1	0	0	0
PUSO	0	0	0	1	0	0
ANILLO	0	0	0	0	0	1

Tabla 1: Ejemplo representación one-hot

Las principales ventajas de esta representación son la facilidad de implementación y la simplicidad en la corrección de errores. La principal desventaja es que tiene un coste espacial muy elevado, ya que el tamaño de la matriz crece en función del vocabulario introducido. Además, de que los vectores resultantes son muy dispersos. Finalmente, la representación *one-hot* es incapaz de capturar las relaciones temporales y semánticas ente palabras.

3.3. Bolsa de palabras

El método Bolsa de palabras o *bag of words.*es una técnica muy usada, [13] es uno de los primeros documentos que hace referencia al termino. *BoW* es un método de representación de documentos, donde para cada documento se crea un vector de talla del vocabulario $\mathrm{x} = \{x_1, x_2, \ldots, \mathrm{x}_n\}$. Cada componente x_i hace referencia a una palabra i en el vocabulario. En este modelo el valor de dicha componente puede ser calculada a partir de los vectores *one-hot* de las palabras en una frase y puede representar diferente información:

- Existencia de la palabra en el documento (representación binaria). $x_i = 1\ si\ i \exists\ en\ doc$. Lo que es equivalente a hacer una operación "*and*" sobre todos los vectores *one-hot* de un documento.

- Frecuencia de la palabra en el documento. En la *ilustración 4* observamos el cálculo de la frecuencia de las palabras del vocabulario para tres documentos distintos. Lo que es equivalente a hacer un sumatorio sobre todos los vectores *one-hot* de un documento

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Ilustración 4: Cálculo *bag of words* con frecuencia

- Calculo ponderado de Tf-idf. Las ecuaciones se muestran en la *ilustración 5.* Siendo $f(t,d)$ la frecuencia del término t en el documento d. Este cálculo permite obtener una representación más adecuada de los documentos.

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(t,d) : t \in d\}} \quad \rightarrow \text{Fórmula TF}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad \rightarrow \text{Fórmula IDF}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D) \quad \rightarrow \text{Fórmula TF*IDF}$$

Ilustración 5: Fórmulas Tf-idf

El modelo *BoW* es muy útil en contextos donde es importante los valores de frecuencia de las palabras. Además, aunque este método no resuelve el problema de relaciones temporales entre las palabras, esto se puede solucionar mediante el uso de *n*-gramas. Finalmente, esta representación reduce el coste espacial respecto a la representación matricial *one-hot* aunque puede seguir siendo muy elevado.

3.4. Word embeddings

Los *word embeddings* surgieron como alternativa a las representaciones discretas detalladas en las secciones 3.2 y 3.3. Estos representan, mediante vectores de reducida dimensionalidad, algunas relaciones semánticas entre las palabras. Permiten conservar propiedades lingüísticas como el género y tiempo verbal, entre otras. En la *ilustración 6* se observa varios ejemplos de estas relaciones. Se podría obtener el vector "*walked*" de la siguiente forma: V("*swimming*") + V("*swam*") – V("*walking*") = V("*walked*"). También se podría obtener el vector Madrid según: V("*Italy*")+V(Rome)-V("*Spain*")=V("Madrid").

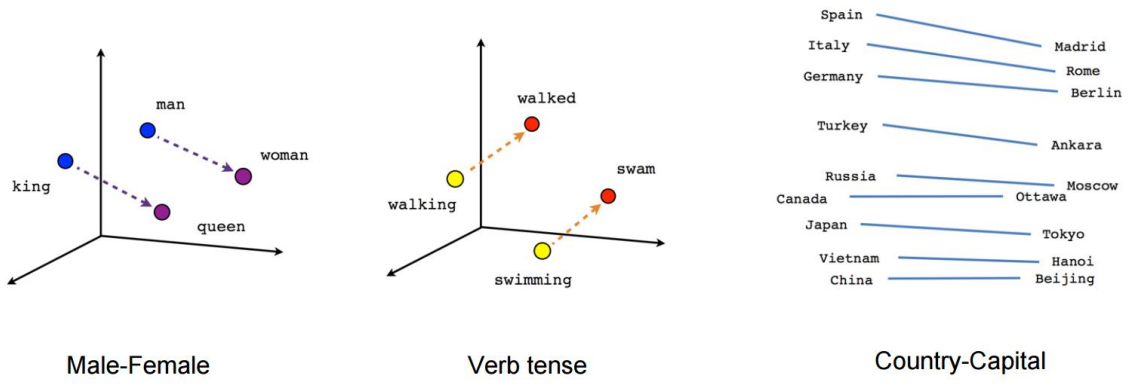

Ilustración 6: Ejemplo relaciones lingïsticas de word embeddings

Los modelos de word embeddings están basados principalmente en redes neuronales. Los modelos contextuales se caracterizan por representar las palabras de forma diferente, dependiendo del contexto en el que aparecen. Estos modelos son más recientes y tienen mayor complejidad. Algunos ejemplos de algoritmos son: BERT [14] y ELECTRA [15].

Los modelos incontextuales, representan mediante un único vector todas las acepciones y relaciones semánticas de una palabra, entre los cuales están *FastText* [16] y *Word2Vec.* Nos centraremos en este último, ya que es el usado en este proyecto.

Finalmente veremos algunas técnicas de colapsado de *embeddings,* que permiten obtener, mediante fórmulas sencillas, vectores representativos de unidades mayores: documentos, párrafos, temas, etc., basados en los vectores de las palabras que los forman.

3.4.1. Modelos contextuales

Los modelos contextuales generan vectores de representación distintos dependiendo del contexto de las palabras. A continuación, se comentan brevemente dos modelos estado del arte (BERT y ELECTRA):

BERT [14], *Bidirectional Encoder Representations from Transformers*, aprende representaciones bidireccionales y contextuales a partir de texto no etiquetado, teniendo en cuenta el contexto derecho e izquierdo.

El proceso de entrenamiento BERT se realiza en dos fases como se muestra en la *ilustración 7*. Primero se realiza un preentrenamiento "*self-supervised*" sobre texto no etiquetado, enmascarado tokens de forma aleatoria para aprender un "*masked language mode*l". A continuación, en *fine-tuning* se parte de los parámetros obtenidos tras el preentrenamiento y se entrena con texto supervisado para la tarea objetivo. Este proceso permite representar bidireccionalmente las palabras dependiendo del contexto en el que aparezcan.

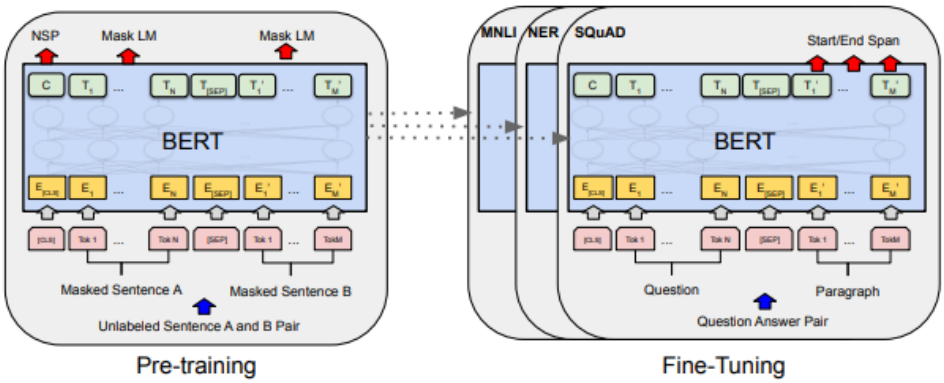

Ilustración 7: Fases algoritmo BERT [14]

ELECTRA [15], *Efficiently Learning an Encoder that Classifies Token Replacements Accurately*, está compuesto por dos modelos más sencillos: el generador, que corrompe el texto de entrada, y el discriminador, que discierne palabra a palabra, si una palabra ha sido reemplazada por el generador, es decir, el discriminador resuelve un modelo de lenguaje enmascarado, de forma que actúa como un clasificador. Esto permite obtener un modelo más "*sample-efficient*" que BERT según [15].

Este método, de la misma manera que BERT, permite representar bidireccionalmente las palabras dependiendo del contexto en el que aparezcan.

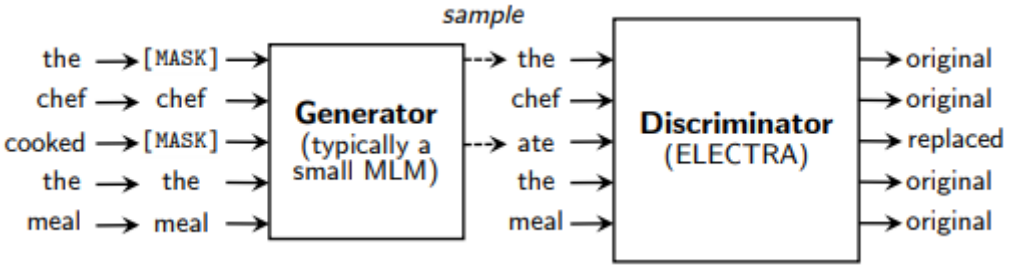

Ilustración 8: Modelo entrenamiento ELECTRA [15]

3.4.2. Modelos Incontextuales

Los modelos incontextuales tratan de representar el mayor número de relaciones sintácticas y semánticas de cada palabra mediante vectores únicos, es decir, usando un único vector para representar todos los contextos posibles en los que ha aparecido una palabra. Hay muchos modelos *word embeddings* incontextuales, algunos de los más conocidos son: *Word2Vec*, *Glove* y *FastText*.

Tomas Mikolov propuso el algoritmo *Word2Vec* en [17] y [18] , haciendo uso de arquitecturas como *CBOW* y *Skip-gram* combinadas aproximaciones eficientes de la función *softmax*, como *hierarchical softmax* o *negative sampling*. *Word2Vec* no es un algoritmo único sino una forma continua de representar mediante redes neuronales las relaciones entre palabras. A continuación, se presentan dos arquitecturas posibles para la creación de modelos *Word2Vec*.

El objetivo de la arquitectura CBOW, *Continuous Bag Of Words*, es predecir el vector de la palabra w(t) conociendo su contexto en una ventana de k elementos: $\{w(t-k), \dots , w(t-1), w(t+1), \dots, w(t+k)\}$ como se muestra en la *ilustración 9*. Cada entrada $w(t)$ es un vector *one-hot.* Es decir, predice la palabra dado el contexto $\left(p\left(w(t)\right)\middle|\{w(t-2), w(t-1)w(t+1)w(t+2)\}\right)$ lo que provoca que se modifiquen los pesos de la red de *embeddings*.

Este modelo suaviza la distribución de probabilidad [17] por lo cual puede no capturar tantas propiedades lingüísticas como *skip-gram,* por lo que suele comportarse mejor en situaciones donde este suavizado es beneficioso para la representación.

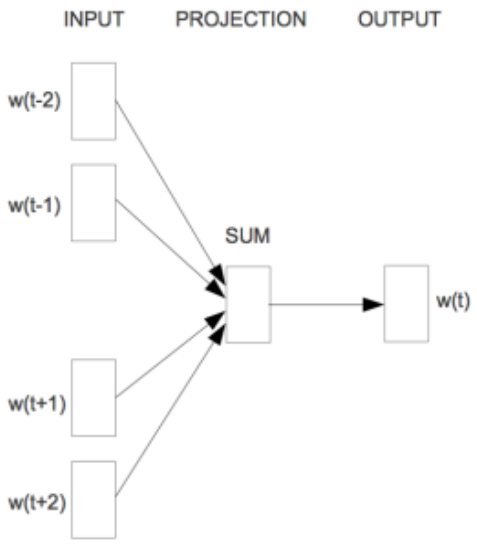

Ilustración 9: Arquitectura CBOW con contexto múltiple

La arquitectura *skip-gram* trata de predecir el contexto de una palabra w(t). Dada una secuencia de muestras de entrenamiento $\{x_1, x_2, \dots, x_n\}$ compuesta de las palabras $\{w_1, w_2, \dots, w_T\}$ se trata de maximizar el logaritmo de la probabilidad: $\frac{1}{T}\sum_{t=1}^{T}\sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j}|w_t)$, siendo T la longitud de la secuencia de entrenamiento y c el tamaño del contexto. Es decir, dado el vector de representación de una palabra, trata de predecir palabras cercanas al mismo.

Si el valor C usado para el cálculo es elevado, los ejemplos de entrenamientos son más completos permitiendo generar vectores más completos, lo que provoca que esta arquitectura se comporte bien ante un corpus ante corpus con un número elevado de muestras de aprendizaje, ya que es capaz de sacar más información que CBOW. A pesar de esto la arquitectura *skip-gram* tiene un coste temporal mayor que CBOW, pero se puede hacer uso de otras técnicas para reducir la complejidad de su cálculo [12].

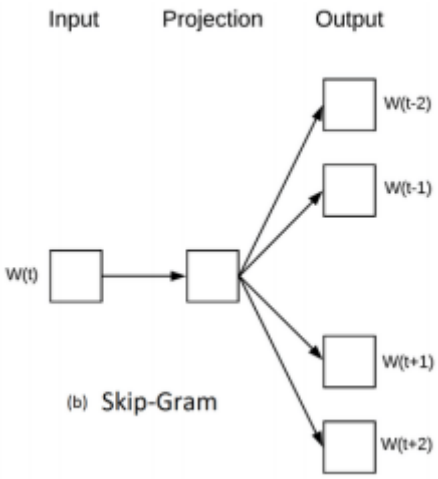

Ilustración 10: Arquitectura Skip-Gram con contexto múltiple

3.4.3. Colapsado de embeddings

Hasta ahora hemos tratado representaciones vectoriales de palabras. Para representar documentos o secuencias lingüísticas a partir de los *word embeddings* de las palabras que lo forman se pueden usar técnicas de colapsado de *embeddings* [12]. En nuestro caso lo usaremos para representar los temas usados en el modelo de la aplicación. Algunos de ellos son:

- Suma: Consiste en sumar todos los vectores que forman parte del documento para obtener su representación vectorial: $V(doc) = \sum V(w_i)$. Siendo "V" la representación vectorial de la palabra o documento dado. De esta manera el documento es representado como la suma semántica de sus constituyentes. En el proyecto usamos esta representación para el cálculo de los *embeddings* de los *tweets*.
- Centroide: Está basado en el modelo suma, donde la suma de vectores de los constituyentes que forman el documento se normaliza por el número de constituyentes. En nuestro proyecto usaremos este modelo para obtener los *embeddings* de los temas y cuentas de *Twitter*.

El colapsado de *embeddings*, al igual que la representación mediante Bolsa de palabras, provoca la perdida entre las relaciones temporales, de orden, de las palabras.

4. Identificación del problema y propuesta de solución

En este capítulo se presenta, a grandes rasgos, a qué trata de dar respuesta el proyecto. Se empieza describiendo el problema sobre el que se ha realizado el proyecto. A continuación, se describe nuestra solución, su diseño y la arquitectura propuesta. Finalmente, se describen las tecnologías utilizadas para el desarrollo de la aplicación.

4.1. Identificación del problema

En las redes sociales hay gran diversidad de cuentas de usuarios, cada una escribe y muestra los distintos temas en los que tiene interés mediante sus publicaciones. En el caso de *Twitter*, estas publicaciones son principalmente texto (*tweets*) que pueden enlazar contenido multimedia. El gran volumen de datos puede implicar que para los usuarios particulares o empresas sea difícil buscar cuentas afines a sus intereses o temas.

Twitter puede ser considerado como un servicio de micro blogs, puesto que los *tweets* no pueden tener más de 280 caracteres. Encontrar cuentas de interés mediante estos *tweets* puede resultar arduo, especialmente si se requiere procesar grandes cantidades. En nuestro proyecto se intenta automatizar la clasificación, estudiando la semántica de los *tweets*.

El proyecto trata de dar respuesta a la necesidad de clasificación de cuentas de redes sociales en temas, haciendo uso de los textos publicados en las mismas. Esto podría ser usado para diferentes objetivos como: buscar usuarios de *Twitter* con gustos semejantes al de un usuario dado, saber si ciertos usuarios están interesados en algún tema en particular, marketing, etc.

4.2. Propuesta de solución

Se propone el desarrollo de una aplicación centrada en dos partes: En primer lugar, un modelo de clasificación que permita identificar la temática de un usuario de *Twitter*, en función de sus publicaciones. En segundo lugar, una interfaz gráfica amigable, fácil de usar y que permita la visualización de los resultados de forma clara.

La aplicación debe de permitir, a partir de cualquier cuenta de usuario de *Twitter, tweet* o texto introducido directamente en la aplicación, identificar los temas, cuentas y *tweets* más similares de la base de datos. Así mismo debe presentar estos datos de una forma amigable y clara para el usuario.

Para el modelo de clasificación de texto se propone el uso *word embeddings* incontextuales, con el objetivo de representar los temas, cuentas de usuario y *tweets*. Se usará la representación *Word2Vec* para cada termino, dado un modelo preentrenado proporcionado por los tutores del proyecto. Dicho modelo ha sido entrenado con 87 millones de *tweets* en español. Hace uso de una arquitectura similar a la usada en [19], pero con una dimensionalidad menor debido a la diferencia entre el número de muestras usadas para entrenar el modelo. Se trata de un modelo *skip-gram* de 300 dimensiones, que ha demostrado un buen comportamiento en trabajos previos [20].

Partiendo de este modelo se representan los *tweets* mediante colapsado de *embeddings*. Tras lo que se representan las cuentas en función de los *tweets* de cada cuenta. A continuación, se representan los temas a partir de los *tweets* de las cuentas etiquetadas de forma manual en cada tema. Estas representaciones permiten obtener un modelo de clasificación de textos para su posterior uso en la interfaz gráfica.

Finalmente, utilizaremos una estructura de árbol para almacenar el modelo resultante y para obtener los k resultados más similares al tema, cuenta o *tweet* dado por el usuario.

Posteriormente, para evaluar la calidad del modelo, se hace uso del valor Macro-F1 mediante el cálculo de la similitud coseno entre los *word embeddings* de las cuentas de test y de los temas. Cada cuenta se considera clasificada en el tema que presenta máxima similitud.

4.3. Diseño de la solución propuesta

En este apartado se presenta la arquitectura propuesta, la separación de ficheros usada en la solución y la función de cada uno, además de las relaciones y dependencias entre los mismos.

La arquitectura propuesta para la aplicación hace uso los de cinco elementos principales, que se muestran en la *ilustración 11*. Las interrelaciones entre los distintos elementos se representan mediante flechas. Cada flecha indica que el elemento del que parte la flecha hace uso del elemento en el que finaliza la punta de la flecha. Las relaciones entre las diferentes partes de la arquitectura de la aplicación suceden de la siguiente forma:

- **GUI:** Accede a la lógica de la aplicación para disponer a toda la información que se desea mostrar.
- **Lógica de la aplicación:** Accede a *Twitter* para recuperar la información (*tweets*) indicados por el usuario. A continuación, accede al modelo de clasificación para clasificar el texto deseado por el usuario. Finalmente, accede a la base de datos para dar la información pertinente de los datos almacenados.
- **Base de datos:** Accede a *Twitter* para recuperar todos los *tweets* de las cuentas deseadas para la creación y testeo del modelo de clasificación.
- **Modelo de clasificación:** Accede a la base de datos para poder crear el modelo y testear su correcto funcionamiento.

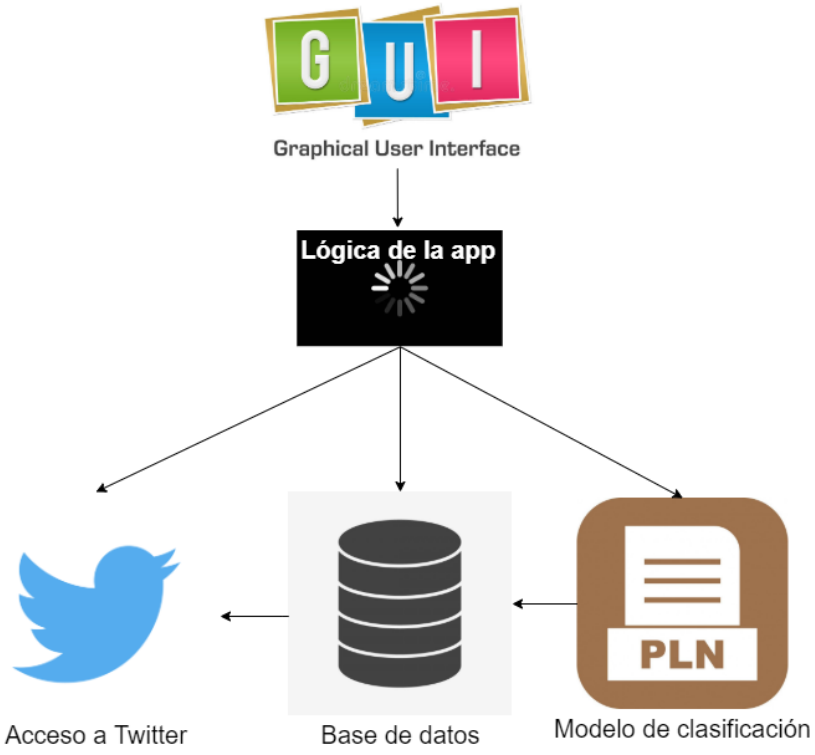

Ilustración 11: Arquitectura de la aplicación

Los ficheros que se describen a continuación son los que permiten las relaciones y funcionalidades descritas anteriormente, y que garantizan el correcto funcionamiento de la aplicación:

- ***Accounts.py*:** Un archivo que define las relaciones entre las cuentas y los temas que tratan, además indica si son usadas para el test o el modelo. También tiene algunas funciones adicionales que permiten conocer más datos sobre las relaciones, como obtener el número de cuentas de test. Todos los elementos de la arquitectura hacen uso de este módulo.
- ***Get_tweets_accounts.py*:** Se utiliza para crear la base de datos. Basándose en *accounts.py,* se recuperan los *tweets* de las cuentas de cada tema, y se añade el tema al que pertenecen al objeto *json* recuperado. Finalmente, se almacenan en la base de datos. En este fichero se crea la relación entre la base de datos y el acceso a *Twitter*.
- ***Get_data_tweets.py*:** En este archivo se han realizado todas las pruebas para la creación del modelo de clasificación del proyecto. Además, sirve para recuperar la información de los datos de la base de datos (número de *tweets* por cuenta y tema), y testear el valor Macro-F1 del modelo. Depende directamente de la base de datos creada previamente y *accounts.py*. Este fichero relaciona los test del modelo con la base de datos.
- ***Indexer.py*:** Sirve para crear y almacenar el modelo de clasificación en una estructura de datos en forma de árbol, para su posterior uso en la aplicación de usuario. Usa la base de datos creada previamente y *accounts.py*. Este fichero relaciona el modelo con la base de datos.
- ***Searcher.py*:** Crea toda la funcionalidad de la aplicación, búsqueda de *tweets* o cuentas, clasificación, etc. Hace uso de los archivos creados en *indexer.py*, y de la base de datos para recuperar la información de los *tweets*

almacenados. En este fichero se describen todas las relaciones de la lógica de la aplicación con los otros elementos de la arquitectura (base de datos, acceso a *Twitter*, y el modelo de clasificación).

- ***App.py***: Basado en la clase *searcher.py,* crea la interfaz gráfica de usuario, permitiendo realizar consultas de forma agradable y visual. En este fichero se introduce la relación entre la GUI y la lógica de la aplicación.

4.4. Tecnologías utilizadas

En este apartado se introduce las tecnologías utilizadas en el desarrollo del proyecto, y la razón de su utilización. Se empieza describiendo el lenguaje de programación usado, junto a las librerías del mismo. A continuación, se menciona el tipo de base de datos usada y finalmente una herramienta de control de versiones.

4.4.1. Python

Para el desarrollo de la aplicación, se ha utilizado el lenguaje de programación *Python* [21]. Las razones principales de su uso son las siguientes: *Python* tiene diversas herramientas muy útiles para creación de modelos de PLN y aprendizaje automático, además es un lenguaje muy limpio y claro, el cual tiene un gran número de librerías que simplifican y facilitan el desarrollo de aplicaciones.

El proyecto se ha realizado en el entorno de desarrollo *Pycharm* [22], el cual facilita el uso de herramienta externas y librerías internas de *Python*. Además, simplifica la depuración de errores, y la visualización y ejecución del programa.

Se ha utilizado diferentes librerías de *Python* para los diferentes aspectos de la aplicación desarrollada:

- **Librerías de recuperación y almacenamiento de información:** La recuperación de datos de *Twitter* se ha realizado mediante *Tweepy* [23]. Esta librería facilita el acceso a *Twitter* y simplifica los métodos de recuperación de *tweets. E*l almacenaje de los *tweets* y su clasificación en los diferentes temas usados en la clasificación se ha realizado mediante *PyMongo* [24]. *PyMongo* permite la creación y el uso de bases de datos *MongoDB*. Finalmente, el almacenaje del modelo procesado para la utilización en la aplicación se ha hecho mediante *pickle* [25], que permite la serialización de objetos en binario.
- **Librerías de procesamiento y cálculo:**
    1. Para el preprocesado del texto y el cálculo de los vectores de las palabras de los *tweets* mediante el modelo de representación *word embedding,* más concretamente *Word2Vec,* se ha utilizado *Gensim* [26]. Esta librería tiene las herramientas necesarias para realizar el preproceso de los datos de entrada, y representación vectorial de texto basado en un modelo preentrenado.
    2. Para el cálculo de la calidad de modelo (Macro F1), y la creación de la estructura en la que se han almacenado los vectores (estructura de árbol de K dimensiones), se ha utilizado *scikit-learn* [27]. Para trabajar con los vectores resultantes del modelo se ha usado *Numpy* [28], una librería especializada en la manipulación eficiente de vectores.

- **Librerías de visualización:** En la parte gráfica de la aplicación se ha utilizado *tkinter* [29], la cual es una librería propia de *Python* que permite crear entornos gráficos de manera simple y eficaz. *tkinter* tiene los elementos necesarios y suficientes para el desarrollo de una interfaz gráfica acorde con los requisitos de este proyecto. (GUI).

4.4.2. MongoDB

Para el almacenaje de los *tweets* se ha utilizado la base de datos *NoSQL MongoDB* [30]. Esta librería permite el almacenamiento, modificación y recuperación de datos de forma sencilla y rápida.

Las bases de datos *MongoDB* son bases de datos no relacionales, es decir, almacena los datos de forma flexible. En nuestro caso, los datos no se almacenan de forma estructurada en tablas interdependientes ya que dificultaría su uso. En nuestra aplicación cada *tweet* se almacena en un documento *json*, permitiendo que no todos los documentos tengan los mismos campos y pudiendo realizar búsquedas por cualquiera de estos campos.

4.4.3. GitHub

Git es una aplicación de código abierto que permite el control de versiones de un proyecto, de forma clara, sencilla y facilitando la ejecución de diferentes operaciones como: recuperar antiguas versiones, modificaciones, etc.

GitHub [31] es una plataforma en la nube basada en *Git* que permite mantener un control de versiones en línea, además de compartirlo con otros usuarios y trabajar en paralelo varias personas en un mismo proyecto. En nuestro caso, para facilitar la comunicación y revisión por parte de los directores, se ha creado un proyecto en la plataforma en el siguiente enlace: https://github.com/dierop/TFG.

5. Desarrollo de la solución

En este apartado se describe el proceso seguido, desde la extracción de los datos al diseño de la aplicación. A pesar de la descripción lineal del proceso, este se ha realizado de forma circular. Tras las evaluaciones, se han añadido elementos que mejoran la funcionalidad de la aplicación, la robustez y la calidad.

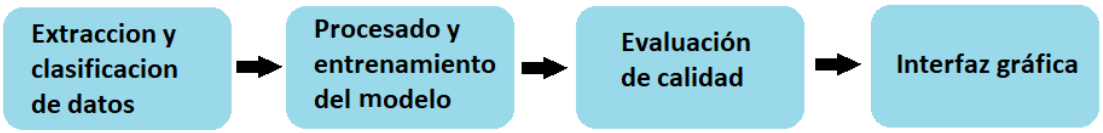

Ilustración 12: Proceso de desarrollo de la solución

El proceso general seguido para la creación del proyecto es el siguiente: En primer lugar, se comenzó buscando cuentas de *Twitter* centradas en los temas a clasificar, se extrajeron sus *tweets* y se clasificaron en el tema correspondiente. A continuación, se preprocesaron los *tweets* y se representaron mediante *word embeddings,* concretamente *Word2Vec*, creando el modelo para su posterior uso. El siguiente paso consistió en evaluar el resultado obtenido mediante el cálculo de la métrica de evaluación (Macro-F1). Tras comprobar su correcto funcionamiento se diseñó una forma eficiente de almacenamiento y recuperación de vectores, haciendo uso de árboles k-dimensionales. Finalmente, se creó una interfaz gráfica que permite determinar la temática predominante de una cuenta y obtener cuentas y *tweets* similares.

5.1. Extracción y clasificación de datos

El primer paso del proyecto consistió en decidir cuáles eran los temas utilizados para clasificar los perfiles. Se decidió utilizar los seis siguientes: *cocina, deportes, tecnología e informática, política, videojuegos* y *otros*. El tema *otros* hace referencia al resto de temas que por su extensión no han podido tenerse en cuenta.

Para la creación del modelo, cada tema queda definido en función de los *tweets* de 30 cuentas distintas. Para su posterior evaluación, utilizamos 12 cuentas adicionales, no solapadas con las que definen los temas. En total se han recopilado 248 cuentas y 629.031 *tweets*. En la *tabla 2*, se presenta el número de *tweets* usados en cada tema para el entrenamiento del modelo y para la evaluación. En el *apéndice 1: cuentas de twitter* se muestran las cuentas de *Twitter* usadas en el proyecto y la cantidad de *tweets* recuperados por usuario.

Temas	Cocina	Deporte	Tecnología e informática	Política	Videojuegos	Otros
Modelo	71.904	77.830	83.191	65.713	80.145	71.863
Test	25.395	31.429	35.042	24.584	32.957	28.978

Tabla 2: Número de *tweets* por tema

Los datos se recuperaron mediante la librería *tweepy,* mediante la cual se obtuvo el histórico de *tweets* de cada cuenta seleccionada. De los *tweets* recuperados por cuenta, eliminamos los *retweets,* ya que es bastante común que las cuentas tengan *retweets* de temas diferentes al que se ha considerado que pertenecen. Los *tweets* recuperados se clasificaron, añadiendo al objeto *json* que los representa*,* un campo "Tema" con el tema al que pertenece el usuario emisor del *tweet*, y finalmente se almacenaron en la base de datos *MongoDB*.

5.1.1. Diversidad en los temas

Para la selección de cuentas se ha considerado la diversidad en los temas seleccionados y el nivel de centralización de la cuenta en el tema, es decir, que las cuentas no traten más temas de los que están clasificadas. Seleccionar cuentas con un alto nivel de centralización en los temas, ha causado que haya una gran cantidad de cuentas de revistas o periódicos, lo que implica que la forma de escritura usada por estas cuentas es bastante estructurada y menos coloquial que una cuenta de un usuario común. Para la diversidad en los usuarios seleccionados se han considerado diversas características según el tema, como: la demografía, el género, los diferente tipos o subtemas que tienen, y personas referentes del tema en concreto.

La selección de las cuentas de test se hizo de forma posterior a la selección de todas las cuentas usadas en el modelo. Esta se realizó de forma aleatoria, seleccionando doce cuentas entre el total de cuentas de cada tema sin mirar sus características particulares, para evitar sesgar los resultados del modelo. A continuación, se van a describir las características específicas respecto a las que se han seleccionado las cuentas de cada tema para lograr diversidad en los temas.

En cuanto a la diversidad en el tema *cocina,* se han elegido principalmente cuentas de recetas libres, es decir no caracterizadas por algún estilo especifico, ya sea demográfico, calidad u otros elementos (*@directopaladar*, *@Gastronomiaycia*). También se han seleccionado cuentas respecto a la demografía local (*@GastronomiaCyL, @andgastrotur*) e internacional (*@ComidasDelPeru*). Además, se han utilizado cuentas que tienen en cuenta otros tipos de alimentación, alergias y veganismo (*@GastronomiaVeg, @gastrosingular*). Finalmente se han elegido alguna cuenta de chefs reconocidos (*@chefghgonzalez*).

La diversidad en el campo del *deporte* ha consistido en introducir distintos tipos de deportes *(@atletismoRFEA*, *@NBAspain*), ya que los periódicos deportivos generales (*@FOXDeportes*, *@marca*) se suelen centrar en el futbol. Además, se ha tenido en cuenta el deporte femenino (*@deporte_mujer*, *@revistalideras*), pues las revistas deportivas no lo tratan en exceso. Se han utilizado cuentas locales (*@valenciacf*) e internacionales (*@Tokyo2020es*, *@Mercado_Ingles*). Finalmente se ha introducido cuentas centradas los propios deportistas (*@yosoynoticia_*).

El tema *tecnología e informática* está compuesto por una gran cantidad de cuentas de tecnología generalistas *(@xataka, @abc_tecnologia, @TecnosferaET*). Además, hay cuentas centradas en distintos aparatos electrónicos como: auriculares, móviles, partes de ordenador, etc. *(@MuyComputerPRO, @ReformaGadgets*). También se han usado cuentas centradas en *software* (@*muylinux*), y personas que realizan críticas de productos tecnológicos recientes *(@_VanCajun, @jmatuk*).

En el ámbito de la *política* se ha tenido en cuenta tres aspectos principales: demografía, género y afiliación política. Respecto a la demografía, se han seleccionado cuentas nacionales *(@MasPais_Es, @LPGPolitica*), regionales *(@CsCValenciana, @susanadiaz*), e internacionales *(@tuvozalmundo, @esglobal_org*). También se han seleccionado políticos importantes de ambos géneros *(@pablocasado_, @IreneMontero*). Además, se han usado cuentas de distintas afiliaciones políticas *(@vox_es, @PODEMOS*). Finalmente, se han seleccionado cuentas centradas en problemas humanitarios globales *(@amnistiaespana, @oxfam_es*).

En el tema *videojuegos*, se ha priorizado las cuentas de información y crítica no especializadas en tipos de consola o juegos (@VandalOnline, @GamereactorES, @Videojuegos). Además, se han escogido algunas cuentas de consolas (@Nintenderos, @Xbox_Spain), empresas de videojuegos (@Ubisoft_Spain), videojuegos populares (@Warcraft_ES, @PokemonGOespana), *streamers* (@TwitchES), y de deportes electrónicos (@marcaesports, @LVPes).

El tema *otros* presenta una variedad de temas y cuentas muy extenso, dentro de los cuales se ha seguido tratando de mantener la diversidad demográfica y de género principalmente.

Se han utilizado cuentas de *influencers* (@twin_melody), shows (@myh_tv), dibujantes (@LadyLauryCandy), viajes (@viajar), historia (@HistoriaDiceQue), cine (@_Cinefilos_), escritura (@Trabalibros), religión (@relibertad) y comedia (@IgnatiusFarray), entre otras.

La gran diversidad de temas y cuentas permite introducir en esta categoría muchas de las cuentas y temas que no se han podido tener en cuenta en el proyecto, intentando abarcar el mayor número posible de temas.

5.2. Modelo de clasificación

Para crear el modelo de clasificación de texto, se han seguido los siguientes pasos: el preprocesamiento del texto, la representación de los temas, cuentas y *tweets* en vectores *Word2Vec*, el modelado mediante colapsado de *embeddings,* y finalmente, el almacenamiento mediante la estructura de datos *K-D Tree*. Estas tareas se hicieron de forma continua para poder almacenar el modelo resultante directamente sin almacenar los pasos intermedios.

5.2.1. Preproceso

El texto de los *tweets* no siempre es correcto, puede tener mayúsculas en posiciones arbitrarias, jergas, *urls*, y caracteres no comunes como: emoticonos, hashtags, arrobas, etc. Estas peculiaridades pueden afectar posteriormente a las técnicas de PLN.

En nuestro caso, se han eliminado los caracteres no alfanuméricos exceptuando los acentos, y convertido todo el texto a minúsculas. Además, para su posterior transformación en vectores, se ha tratado cada palabra como un token.

También hemos eliminado las *stopwords*, lo que ha permitido mejorar notablemente el rendimiento del modelo, ya que son palabras muy frecuentes, compartidas por los temas, que no aportan información discriminante.

Hemos evaluado tres conjuntos de *stopwords* distintos, obtenidos de [32], [33], [34]. Estos conjuntos están compuestos por 137, 308 y 447 palabras respectivamente. Experimentalmente se decidió el uso del último conjunto por su mejor comportamiento en las pruebas realizadas. En el apartado 5.5.2 se puede observar las pruebas realizadas.

5.2.2. Entrenamiento

Para la creación del modelo usado en la aplicación, se ha partido de un modelo *Word2Vec* preentrenado, proporcionado por los tutores del proyecto. Para representar vectorialmente cada token, cargamos el modelo preentrenado mediante la librería Gensim. Esto nos permite obtener el vector de representación para cada palabra. Los vectores obtenidos mediante el modelo *Word2Vec* son de 300 dimensiones.

Mediante operaciones sencillas de colapsado de *embeddings* y partiendo de las representaciones *Word2Vec* de cada palabra de los *tweets* se han calculado los vectores resultantes de cada tweet, tema y cuenta.

La fórmula usada para calcular los *tweets* es la siguiente: $\sum W2V[palabra]$. Siendo *w2v* la representación mediante el modelo *Word2Vec.* Por lo tanto, la representación de un *tweet* es el sumatorio de las representaciones de las palabras que contiene.

La representación de las cuentas ha sido calculada mediante la fórmula: $\sum W2V[tweet]$. Los *tweets* usados para la representación de cada cuenta son los escritos por su usuario.

La representación de los temas se ha realizado mediante la misma fórmula que en las cuentas, $\sum W2V[tweet]$, pero los *tweets* usados para la representación son todos los pertenecientes a las cuentas etiquetadas en cada tema.

5.2.3. Estructuración de los datos del modelo

Tras la obtención de los vectores que representan cada elemento del modelo, para su posterior acceso eficiente desde aplicación, se usó la estructura de datos *K-D Tree* o árbol k-dimensional, utilizando la librería *scikit-learn,* Este árbol k-dimensional permite obtener los vectores más cercanos a uno dado. En nuestro caso la dimensión hace referencia al tamaño de los vectores de representación (vectores de dimensión 300).

Un árbol k-dimensional es un árbol de búsqueda binario donde cada nodo es un punto de k dimensiones en el espacio. El funcionamiento del árbol para representar los datos es que en cada nivel se compara una dimensión de la representación, separando los nodos hijos respecto al valor del nodo dado en la dimensión evaluada en el nivel. En la *ilustración 13* se observa el proceso de creación de un árbol k-dimensional de dos dimensiones, y las separaciones en el espacio que crea cada punto. Por ejemplo, el nodo raíz (30,40) separa en la dimensión x, por lo que todos sus hijos derechos son mayores que 30, y todos los izquierdos son menores.

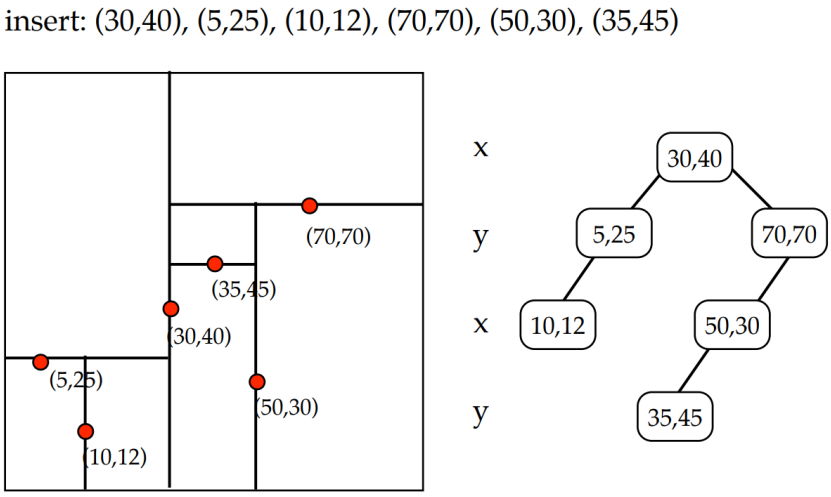

Ilustración 13: Ejemplo de kd tree de dos dimensiones

En nuestro caso el objetivo de usar esta estructura de datos es buscar los k vecinos más cercanos, se seleccionó esta estructura por su bajo coste temporal. En el peor caso buscarlos puede costar $O(n)$, pero en la práctica el coste es cercano a $O(2^d + \log n)$,lo que supone un coste bastante reducido, lo que implica una menor latencia en la búsqueda de los temas, cuentas y *tweets* más similares en la aplicación final.

Se creó un modelo *árbol* k-dimensional separado para cada conjunto de *tweets*, cuentas y temas, estos se almacenaron mediante la librería *pickle*. Para poder identificar los índices a que hace referencia cada elemento de los árboles, cada estructura en árbol se almacenó junto con un diccionario que representa los índices e identificadores pertinentes en cada caso: nombre del tema, nombre de la cuenta o id del *tweet*.

5.3. Evaluación

La evaluación del modelo diseñado se ha realizado mediante la métrica Macro-F1. En los experimentos realizados, las cuentas de los usuarios se clasificaron en función de las distancias a los temas considerados por el modelo mediante la similitud coseno. La evaluación se ha realizado partiendo de los vectores creados en el modelo.

El valor Macro-F1 se ha calcula mediante la fórmula: $\frac{1}{\frac{1}{2}\left(\frac{1}{recall}+\frac{1}{precisión}\right)}$. La librería *scikit-learn* hace uso de dos vectores: el real, y el predicho. El vector real hace referencia a la clase real con la que se ha etiquetado, de forma manual, el conjunto de cuentas usadas. El vector predicho se calcula en función de las similitudes entre cada cuenta y todas las clases a clasificar. En la *tabla 4* se muestra para distintas cuentas, las clases predichas y reales. Para el cálculo de la métrica Macro-F1, cada tema se representa por un valor numérico entre 0 y 5, ambos incluidos.

Para el cálculo del vector predicho, se ha usado la similitud coseno $similitud = cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} {B_i}^2}}$, entre pares de vectores (A, B). Se ha calculado la similitud de cada cuenta de test a los seis temas usados en la clasificación, y se ha asignado a cada cuenta el tema que presenta máxima similitud.

A continuación, se presentan ejemplos del cálculo de la similitud coseno entre cuentas test y los temas, para su posterior cálculo de la métrica Macro-F1. En la *tabla 3* se muestran dos aciertos y dos errores de clasificación en orden. La cuenta *@thermorecetas* se clasifica con el tema *cocina* y *@COE_* se clasifica en *deporte*. *@E1Am1g01nf0rma1* se clasifica en *otros* y *@eduardogavin* en *política*, debiendo clasificarse en *tecnología e informática* y *otros* respectivamente. Finalmente, en la *tabla 4* se presenta el resultado que se usaría para el cálculo de la métrica *Macro-F1.*

	Cocina	**Deporte**	**Tecnología e informática**	**Política**	**Videojuegos**	**Otros**
@thermorecetas	0.0901	0.2827	0.2583	0.3052	0.2623	0.2332
@COE_es	0.1609	0.0853	0.1439	0.1005	0.1323	0.0987
@E1Am1g01nf0rma1	0.1864	0.1969	0.1373	0.1655	0.1407	0.1260
@eduardogavin	0.2191	0.1990	0.1941	0.0990	0.2124	0.1359

Tabla 3: Similitud coseno de cuentas test

	@thermorecetas	**@COE_es**	**@E1Am1g01nf0rma1**	**@eduardogavin**
Valor Real	Cocina	Deporte	Tecnología e informática	Otros
Valor Predicho	Cocina	Deporte	Otros	Política

Tabla 4: Clasificación cuentas para el cálculo Macro-F1

5.4. Aplicación

Para el correcto funcionamiento de la aplicación y la posibilidad de desacoplamiento y cambio de la interfaz gráfica, se ha separado la aplicación en dos capas: lógica de la aplicación e interfaz gráfica. La funcionalidad propuesta de la aplicación es, dado un tweet, una cuenta, o cualquier texto que se desee clasificar, presentar los k temas, cuentas y *tweets* más similares, incluyendo en cada caso los datos más relevantes que puedan servir al usuario.

En la *ilustración 14*, se presenta el diagrama del funcionamiento de la aplicación, que se discutirá con más detalle en las siguientes secciones. Los puntos 1, 7 y 13 suceden sobre la interfaz gráfica, y los puntos del 2 al 6 y del 8 al 12 suceden en la capa lógica. Además, los puntos 3 y 5 hacen uso del acceso a *Twitter.* Asimismo*, l*os puntos 4,6 y del 8 al 10 hacen uso del modelo de clasificación. Finalmente, el punto 12 accede a la base de datos.

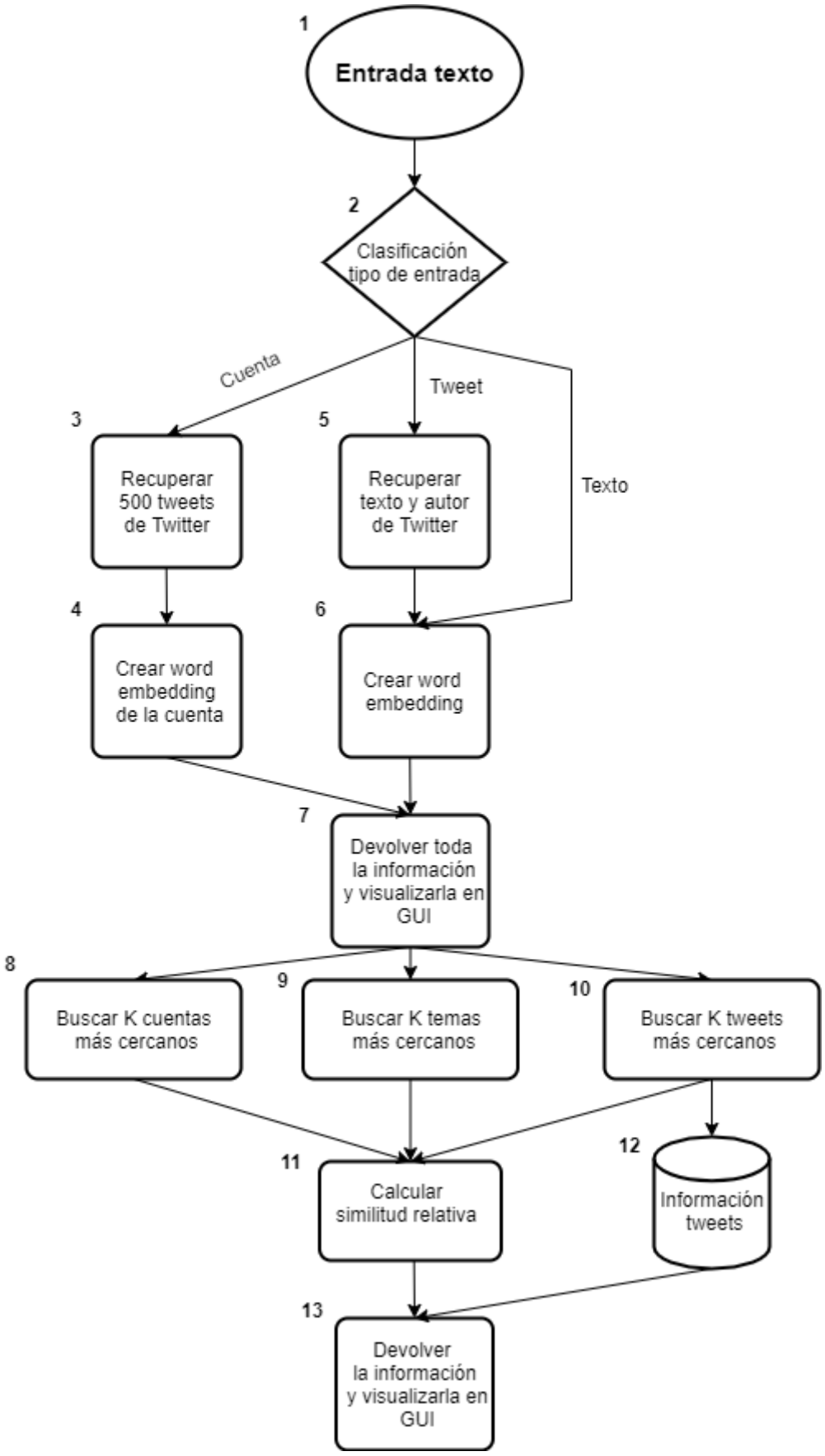

Ilustración 14: Diagrama del funcionamiento de la aplicación

5.4.1. Lógica de la aplicación

La lógica de la aplicación se divide en dos partes. Primero identificamos el tipo de entrada (*tweet*, cuenta o texto), recuperamos la información del mismo si es necesario y calculamos los *word embeddings*. Segundo calculamos las similitudes del vector a los temas, cuentas y *tweets* almacenados en los árboles k-dimensional.

5.4.1.1. Identificación, recuperación de datos y cálculo de word embeddings

Para poder simplificar el uso de la aplicación, se ha automatizado la identificación del tipo de texto de entrada, permitiendo al usuario introducir las diferentes opciones de búsqueda sin necesidad de especificar cual se está usando. Las opciones son las siguientes:

- Búsqueda de *tweets*:

  -URL: https://Twitter.com/**nombre_cuenta**/status/**identificador**

  -**Identificador**

- Búsqueda de cuentas:

  -URL: https://Twitter.com/**nombre_cuenta**

  -@**nombre_cuenta**

- Texto: Siempre que no cumpla con las características previas, en concreto:

  -Más de una palabra

  -Una palabra que no sea como las opciones previas

Teniendo en cuenta que **"nombre_cuenta"** es el nombre de la cuenta en particular que se va a usar, e "**identificador"** es un número, *Twitter* hace uso de números únicos para cada tweet existente, p.ej. 1367102992751751171, el cual es un tweet de la cuenta @pablocasado_.

Una vez identificado de que tipo es la información introducida, pasamos a recuperar los datos de la misma de *Twitter* dependiendo del tipo de entrada:

- *Tweet*: Recuperamos el texto que tiene escrito y su autor.
- Cuenta: Recuperamos 500 *tweets*, para poder calcular su *embedding* posteriormente.
- Texto: No hace falta recuperar información.

A continuación, calculamos el vector *Word2Vec* que los representa, de la misma forma que en cálculo del modelo, y usando el mismo tipo de preproceso (conversión a minúsculas y eliminación de símbolos no alfanuméricos y *stopwords*).

Finalmente, se devuelven cuatro elementos: el autor del texto, el tipo de texto (*tweet*, cuenta o texto), el texto del *tweet* o el texto escrito (en caso de ser una cuenta no devuelve nada), y el vector que lo representa.

5.4.1.2. K Similitudes

Gracias al modelo árbol k-dimensional previamente creado, se pueden devolver los K elementos más similares a un vector de manera simple y rápida.

En nuestro caso, devolvemos en funciones separadas los temas, cuentas y *tweets* más similares ordenados por similitud. Además, se permite en cada caso indicar el número de resultados que deseamos. En cada función se devuelve separando en dos

vectores la información general y el valor de distancia normalizada, la información general dada en cada función distinta es una tupla de:

- Temas: tema.
- Cuentas: cuenta, tema de la cuenta.
- Tweet: cuenta del tweet, tema del tweet, texto del tweet.

En el caso de los *tweets*, puesto que devuelven información almacenada se hace uso de la base de datos.

El valor de similitud normalizada representa el grado de similitud respecto al vector dado en un rango [0,1], permitiendo a los usuarios reconocer si los temas, cuentas o *tweets* iniciales son muy cercanos o si, al contrario, la clasificación está muy clara. Es decir, permite al usuario saber si hay más de un tema similar al texto de entrada.

En la *tabla 5* se ejemplifica la normalización de la similitud, donde texto 1 = "***Una comida equilibrada: revuelto de setas y pure de patatas***", y el texto 2: "***Logramos aprobar la prórroga de los ERTE y las ayudas a autónomos hasta mayo. VOX ha votado en contra de ayudar a autónomos y a los trabajadores que por la pandemia están en ERTE. Que se laven la boca con jabón aquellos cuya única patria es una bandera.***", traído de la cuenta @podemos. Se puede observar cómo el texto 1 y el texto 2 tienen respectivamente varios temas cercanos: *cocina* y *otros*, y solo un cercano: *política*.

	Cocina	**Deporte**	**Tecnología e informática**	**Política**	**Videojuegos**	**Otros**
Texto 1	0	0.93	0.63	1	0.61	0.2
Texto 2	0.99	0.82	0.86	0	1.0	0.9

Tabla 5: Ejemplo distancia normalizada en el ámbito temas

La fórmula usada para el cálculo de la distancia normalizada es, dado el vector de similitudes X , el vector normalizado se calcula para cada valor x_i como $(x_i - \min(X))/(\max(x) - \min(x))$. Obteniendo un resultado normalizado, y teniendo en cuenta que el valor mínimo de distancia es el más cercano al texto de entrada.

5.4.2. Interfaz gráfica

La interfaz gráfica permite al usuario un uso sencillo de la aplicación, permitiéndole visualizar los resultados de la clasificación de una forma agradable y clara.

La interfaz ha sido desarrollada mediante *tkinter*, por su facilidad de uso, claridad y compatibilidad con *Python*. Los principales elementos visuales de la interfaz son *labels*, cuadros de texto, *treeview*, *notebook,* y *grids*.

En la *ilustración 15* se puede observar la interfaz gráfica. Para poder posicionar los elementos de la aplicación de forma clara y precisa, se ha usado un *grid*, el cual permite posicionar los elementos mediante el uso de filas y columnas. Estos elementos son:

- Título de la pestaña (“Buscador *Twitter*”), se ha insertado el logo de *Twitter* a la izquierda del título para indicar que la aplicación hace referencia a *Twitter*.
- Título (“Buscador Semántico *Twitter*”): mediante un *label* de texto, se indica más explícitamente qué tipo de buscador es, además es más llamativo al estar en negrita y tener un tamaño de fuente mayor.
- Texto explicativo (“Clasifica cuentas, *tweets* o texto propio”): mediante un *label* de texto indica qué se puede clasificar en la aplicación.
- Cuadro de texto: Sirve para introducir las cuentas, *tweets*, o texto para su clasificación. Se ha hecho uso de un sistema de eventos para que, por defecto, cuando este está vacío, se indique los diferentes formatos en los que se puede clasificar.
- Botón (“Buscar”): Mediante el botón se realiza la búsqueda del texto introducido en el cuadro de texto.
- Espacio vacío: Debajo del botón y por encima del *notebook*, se ha dejado un espacio vacío. En el momento en el que busquemos nuestra información en este espacio, mediante el uso de *labels* de texto se realimenta la información al usuario. Indicando, cuando procede, el autor y el texto a clasificar (si es un *tweet* o texto).
- *Notebook*: Se ha hecho uso de un sistema de notebook para mantener tres pestañas en el mismo espacio de la aplicación, donde se responde con toda la información deseada. Las pestañas son: temas, cuentas similares y *tweets* similares. En cada una se proporciona información distinta.
- *Treeview*: Dentro de cada pestaña del *notebook,* se ha usado un elemento o *widget* llamado *treeview*. Este sirve para dar información de manera ordenada y clara. Los *treeview* de las distintas pestañas son similares, pero no todos dan la misma información. Se ha usado un *scrollbar* junto al *treeview* para poder visualizar los resultados en caso de que estos ocupen más que el tamaño asignado al mismo.

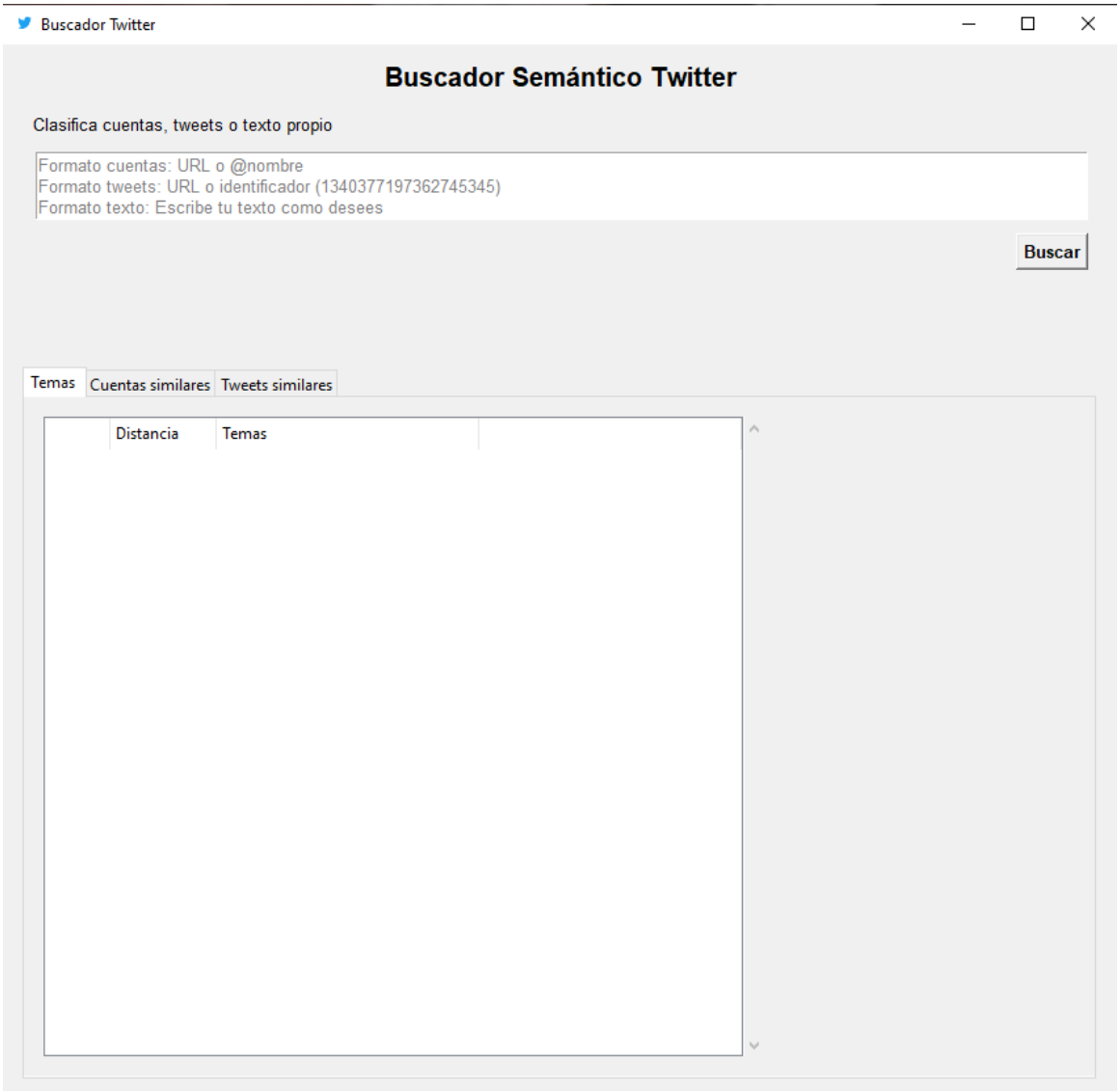

Ilustración 15: Aplicación sin datos introducidos

En las siguientes imágenes se puede observar las diferentes opciones de entrada para el usuario y las diferentes salidas obtenidas. Cada salida observable no está limitada al tipo de entrada, sino que sirve para todas las posibles entradas y viceversa.

En la *ilustración 16* se observa la aplicación al haber introducido la cuenta @pesadillacocina mediante su dirección web https://Twitter.com/pesadillacocina. Cuenta del programa de televisión pesadilla en la cocina, el cual presenta muchas recetas distintas. Observaremos la pestaña “Temas”, que presenta los seis temas de clasificación de forma ordenada, indicando la distancia relativa que tienen a la cuenta. Además, al ser una cuenta de usuario, la única retroalimentación que observamos es el autor.

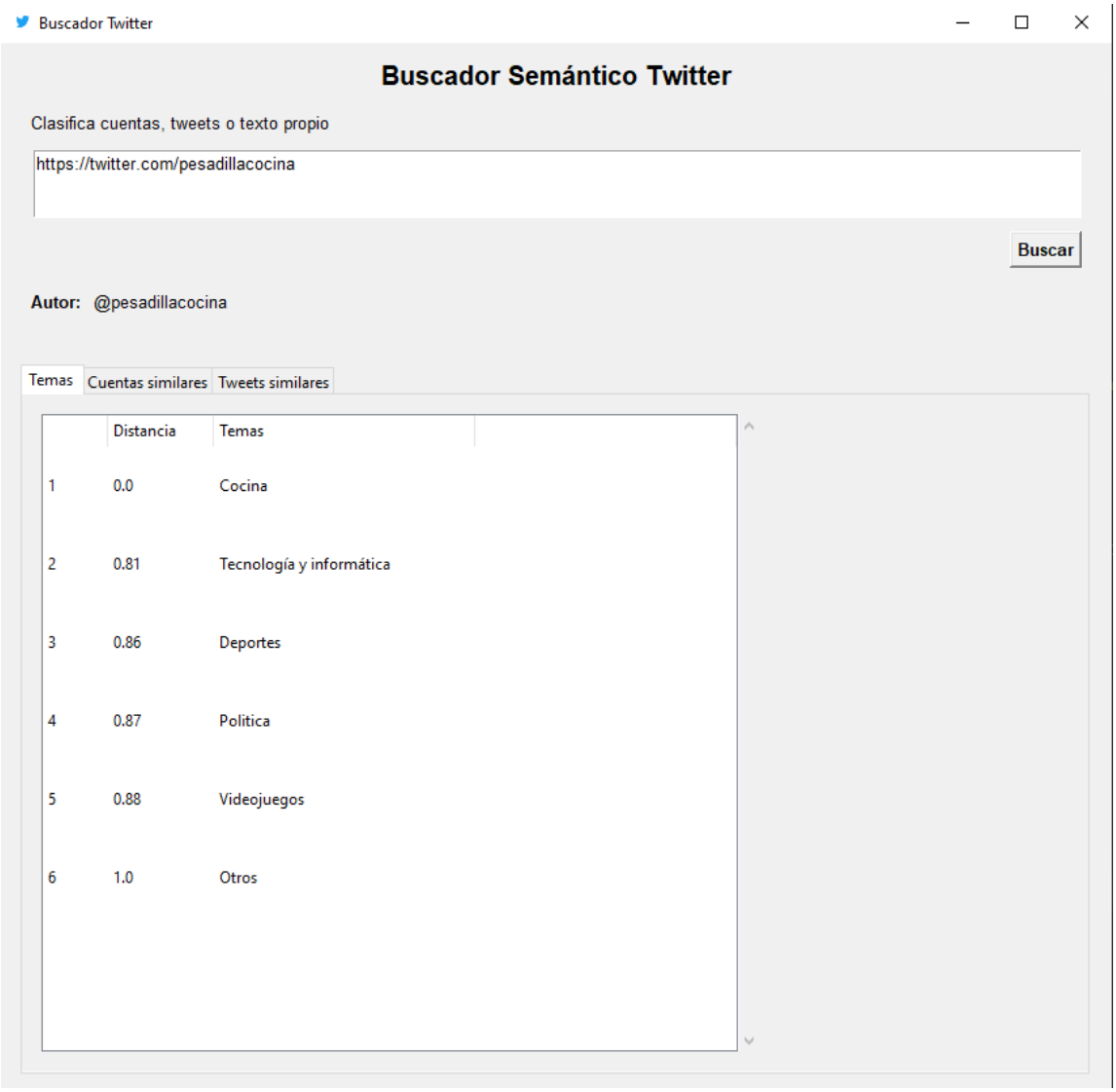

Ilustración 16: Aplicación buscando una cuenta en la pestaña temas

En la *ilustración 17* se muestra la búsqueda de un *tweet* mostrando la pestaña cuentas similares. El *tweet* usado trata un juego popular: Zelda. Se ha introducido mediante su URL: https://Twitter.com/ZarkhosLair/status/1363437109504270336, pero también se podría haber buscado mediante su ID: 1363437109504270336.

En la zona de retroalimentación se muestra el autor, en este caso @ZarkhosLair, y el contenido del *tweet*, en el apartado texto. La pestaña cuentas similares presenta las 8 cuentas más similares al *tweet,* indicando su orden, distancia relativa, nombre y finalmente el tema en el que está considerado esta cuenta. En nuestro caso, los cuatro primeros resultados hacen referencia a cuentas de videojuegos, que es la temática principal del *tweet* a clasificar.

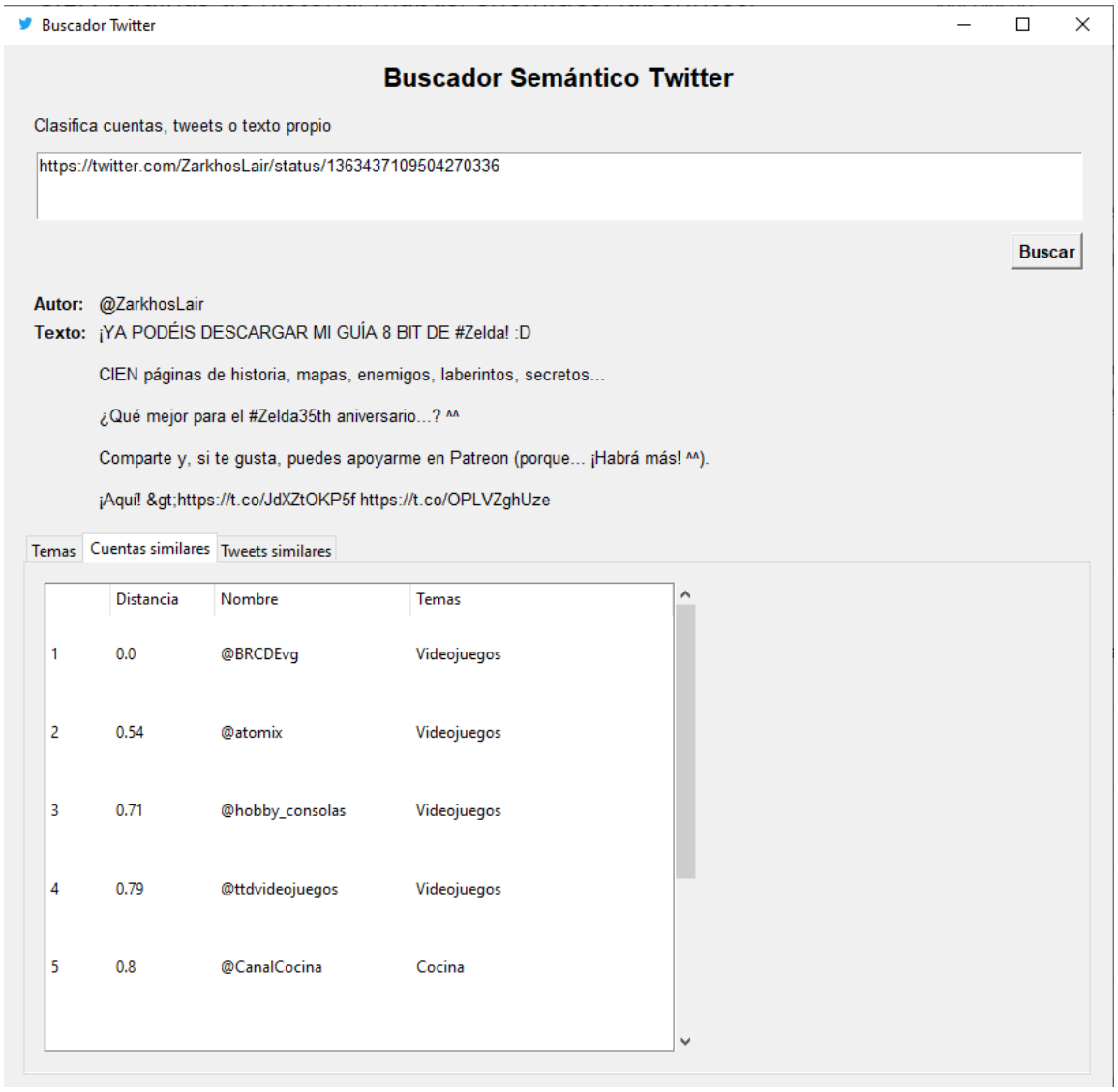

Ilustración 17: Aplicación buscando un tweet en la pestaña cuentas similares

En la *ilustración 18* se muestra la aplicación, usando el texto: *"Al montar un ordenador es importante el balance entre sus distintas componentes: procesador, tarjeta gráfica, memoria RAM, disco duro, etc."* como entrada a la búsqueda, y la pestaña *tweets* similares.

Como en este caso no se está buscando en *Twitter* el texto a clasificar, el autor del texto se muestra como "propio". En este caso, también se muestra el texto clasificado. La pestaña *tweets* similares muestra los 10 *tweets* más similares al texto de entrada.

En cada uno se muestra su orden, distancia relativa, nombre o autor del *tweet,* tema en el que se considera que pertenece y finalmente el texto del mismo. En el ejemplo los seis primeros resultados pertenecen a "tecnología e informática", lo cual se considera el mismo tema que el texto de entrada. Además, se puede observar que los textos de los *tweets* son bastante similares al que hemos introducido.

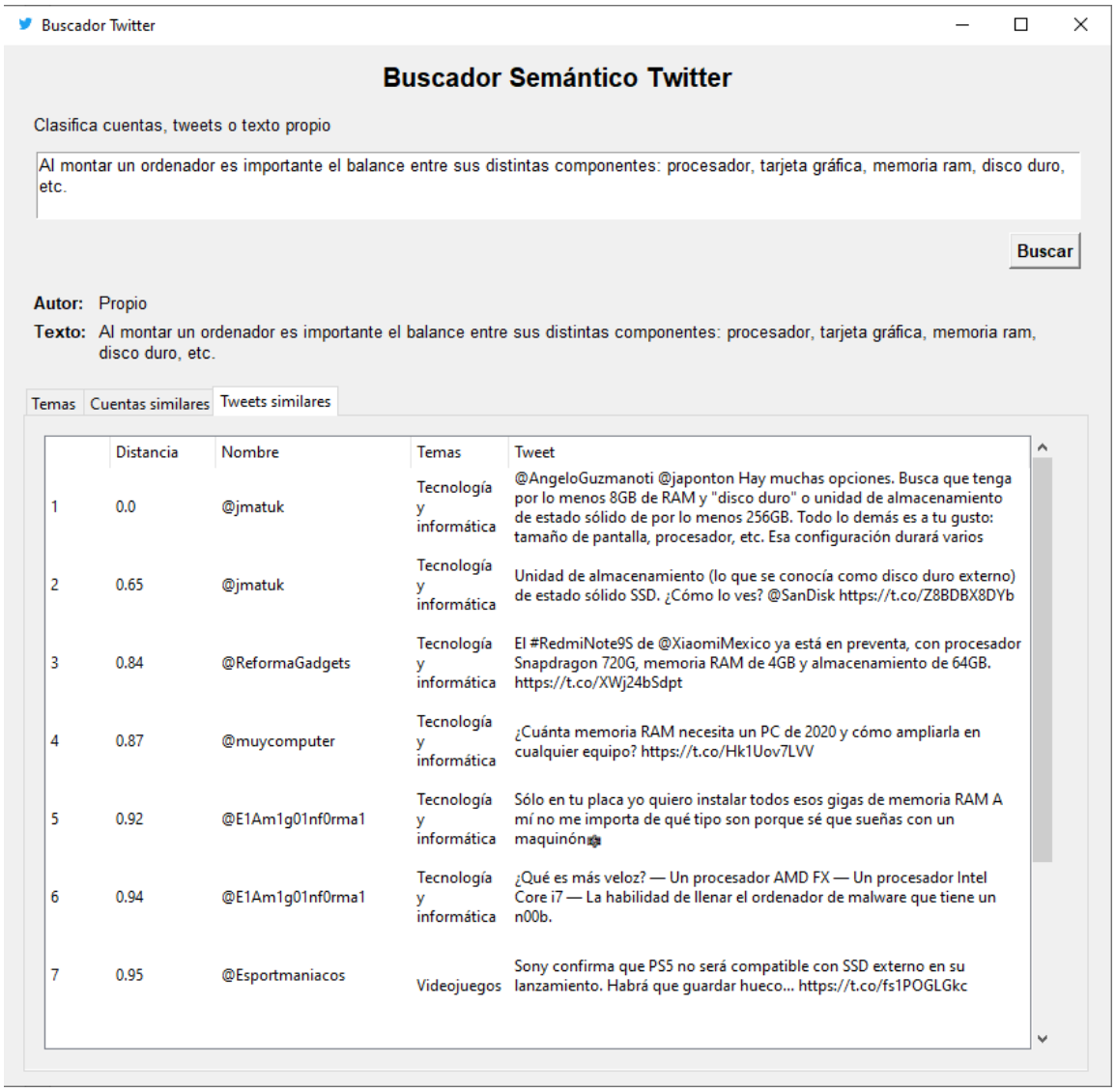

Ilustración 18: Aplicación buscando texto propio en la pestaña *tweets* similares

5.5. Experimentos adicionales

A lo largo del desarrollo de la aplicación, se han probado distintos cambios que afectan a los resultados obtenidos por el modelo. Estos cambios se han realizado progresivamente con la intención de mejorar la calidad de la aplicación. Los cambios realizados pasan desde añadir temas y cuentas, para incrementar la diversidad, a probar diferentes listas de *stopwords*. Además, se han probado diferentes fórmulas para el colapsado de *embeddings*, y distintas estructuras para almacenar el modelo.

5.5.1. Crecimiento del número de cuentas y temas

Al inicio de las pruebas solo se consideraron 5 temas, sin tener en cuenta el tema *otros*. Además, solo había 6 cuentas por tema para la creación del modelo y 5 cuentas por tema para la evaluación. Lo que proporcionaba resultados muy altos, en la métrica Macro-F1, pero muy poco robustos debido a la reducida diversidad.

Numero cuentas del modelo	Valor Macro-F1 sin stopwords	Valor Macro-F1 con stopwords
5	0.868	1.0
30	0.6682	0.8460

Tabla 6: Evaluación del número de cuentas del modelo

Como se observa en la *tabla 6*, los resultados con menor número de cuentas tienen un valor Macro-F1 superior, pero esto implica, por el menor número de cuentas, que es mucho menos robusto al cambio. Además, al no considerarse el tema *otros,* en caso de encontrarse con un tema no considerado, siempre cometería errores. En cambio, con 30 cuentas, el resultado es inferior, pero el modelo es más robusto.

5.5.2. Stopwords

Como se ha mencionado en el apartado 5.2.1, el número de *stopwords* eliminadas afecta a los resultados obtenidos. Para poder decidir qué conjunto de *stopwords* utilizar, se probaron experimentalmente tres, con distinta talla de vocabulario. Experimentalmente se observó que, a mayor número de palabras, mejor métrica Macro-F1 se obtenía. Se nota un mayor cambio al introducir las *stopwords* y al pasar de 137 a 308 *stopwords* que al pasar de 308 a 443. En este último caso el cambio es muy poco notable. En la *tabla 7,* se reflejan los resultados obtenidos aplicando los distintos conjuntos de *stopwords* al modelo de clasificación con 30 cuentas.

	sin *stopwords*	137 *stopwords*	308 *stopwords*	443 *stopwords*
Macro-F1	0.6682	0.7480	0.8347	0.8460
Accuracy %	66.66	73.61	83.33	84.72

Tabla 7: Evaluación del impacto de las *stopwords* en el modelo

5.5.3. Métodos de colapsado de embeddings

Las técnicas de colapsado de *embeddings* se han empleado para el cálculo de los vectores de representación de los *tweets*, cuentas y temas. Además de las técnicas descritas en 5.2.2 se han probado otras, las cuales se han descartado pues pueden afectar de forma negativa al modelo resultante.

Las fórmulas de colapsado de *embeddings* descartadas son muy similares a las usadas, pero los vectores de los temas se calculan como una media aritmética de los vectores de las cuentas que los forman. Esto provoca un aumento del número de divisiones para normalizar, lo que podría producir pérdidas de información en las distintas representaciones.

$$\text{Tweet} = \sum_{i=1}^{N} \text{W2V}[\text{Palabra}_i]$$

$$\text{Cuenta} = \frac{\sum_{i=1}^{N} \text{W2V}[\text{Tweet}_i]}{N}$$

$$\text{Tema} = \frac{\sum_{i=1}^{N} \text{W2V}[\text{Cuenta}_i]}{N}$$

Ilustración 19: Fórmulas de colapsado de embeddings desechadas

5.5.4. Almacenamiento

Para almacenar las representaciones de los temas, cuentas y *tweets* usados en la aplicación, previamente a la utilización del modelo árbol k-dimensional descrito en la sección en *5.2.3,* se probó una base de datos *MongoDB*. Esta opción se desestimó rápidamente por dos razones:

- Incompatibilidad de formatos: El cálculo de los *word embeddings* se ha realizado mediante la librería *numpy,* usando vectores con los datos en *float32*. En cambio, *MongoDB* usa el formato *double.* En el cambio de formato mediante *Python* se ha observado perdida de precisión en los datos.
- Velocidad de recuperación y cálculo: La recuperación de datos en *MongoDB* no es eficiente sin indexación, tras recuperar los datos, tendríamos que comparar cada vector almacenado con el resultante del texto introducido en la aplicación y obtener los k vectores más cercanos. Esto implica un coste temporal excesivamente elevado, a causa de su elevado número (600.000).

6. Conclusiones

Como conclusión se puede afirmar que se han cumplido todos los objetivos planteados. Se ha desarrollado un modelo que permite clasificar *tweets* y cuentas de *Twitter*, en distintos temas, de forma eficiente y con resultados prometedores. Este modelo se ha integrado en la capa lógica de una aplicación

También se ha creado una base de datos con más de 600.000 *tweets,* obtenidos y clasificados en función del tema al que pertenece la cuenta que ha escrito cada *tweet*. Esta base de datos es empleada por el modelo de clasificación para determinar la temática de una cuenta o tweet.

El modelo de clasificación desarrollado está compuesto por seis temas diferentes obteniendo una macro-F1 de 84 puntos, una cifra bastante satisfactoria. Este modelo se basa en la representación vectorial, calculada mediante *word embeddings*, de *tweets*, cuentas y temas.

Por último, se ha desarrollado una interfaz gráfica que permite la visualización de datos clara, y un uso sencillo para el usuario. Esta interfaz permite utilizar el modelo de clasificación para clasificar cuentas, *tweets,* y texto directamente. El resultado obtenido tras la clasificación son los temas, cuentas y *tweets* más similares a la entrada introducida. Además, se ha desacoplado la interfaz gráfica de la lógica de la aplicación, permitiendo realizar cambios sin necesidad de modificar toda la aplicación.

En este trabajo, se han puesto en práctica los conocimientos y las bases teóricas adquiridas en el grado de Ingeniería Informática. Además, me ha permitido adquirir conocimientos y técnicas del campo del PLN

6.1. Relación con las asignaturas cursadas

En este punto se va a presentar la correlación entre el proyecto realizado y las asignaturas cursadas en el grado de Ingeniería Informática, que han sido especialmente útiles para la realización del mismo.

En primer lugar, la asignatura "*Bases de datos y sistemas de información*" presenta el concepto de bases de datos y sus diferentes tipos, además de presentar las bases de datos SQL, su metodología y su forma de recuperar la información, lo que ha facilitado la creación y uso de una base de datos no estructurada.

En segundo lugar, las asignaturas "*Sistemas de almacenamiento y recuperación de información*" y "*Algorítmica*", explican diferentes metodologías y algoritmos que permiten almacenar, procesar y recuperar información, textual, de manera eficiente. En este proyecto, estos conocimientos se han usado para la creación del modelo de clasificación y los algoritmos que presentan la lógica de la aplicación.

En último lugar, las asignaturas "*Ingeniería del software*" e "*Interfaces persona computador*", que proporcionan el conocimiento y la experiencia en la creación de aplicaciones y de interfaces respectivamente, han facilitado la creación de una aplicación, separada por capas (lógica e interfaz), que resulta sencilla y amigable para el usuario.

7. Trabajos futuros

A continuación, se exponen varias mejoras diferentes, basándonos en el proyecto realizado, que pueden servir como guía para la realización de trabajos futuros:

- Uso de *word embeddings* contextuales: En la actualidad se está experimentando con el uso de *word embeddings* contextuales, como en BERT. Experimentalmente se ha visto que estas técnicas suelen mejorar los resultados comparados con los modelos incontextuales, por lo que es posible que su uso permita obtener mejores resultados.
- Probar diferentes técnicas de colapsado de *embeddings*: En el cálculo de colapsado de *embeddings* se han usado dos fórmulas basadas en la suma de los vectores de representación. En un futuro se puede explorar diferentes fórmulas para comprobar cómo afecta al modelo creado.
- Creación de una aplicación en la nube: Para mejorar la aplicación creada en este proyecto se puede crear una aplicación semejante en la nube, utilizando el modelo creado como *back-end,* para permitir el uso de la aplicación a diferentes usuarios.
- Aumento de la base de datos: A efectos del tiempo limitado, el número de temas a clasificar es seis, incluyendo “otros”, lo que es un número bastante reducido comparado con toda la información que hay en internet. Ampliar la base de datos para la creación del modelo, tanto en número de temas como en número de cuentas por tema, puede mejorar considerablemente la calidad del modelo y de la aplicación.

Bibliografía

[1] P. Wang, J. Hu, H.-J. Zeng y Z. Chen, «Using Wikipedia knowledge to improve text classification,» *Knowledge and Information Systems,* vol. 19, p. 265–281, 2009.

[2] J. M. Quinteiro-González, E. Martel-Jordán, P. Hernández-Morera, J. A. Ligero-Fleitas y A. López-Rodriguez, «Clasificación de textos en lenguaje natural usando la Wikipedia,» 2011.

[3] L. Liu, L. Tang, W. Dong, S. Yao y W. Zhou, «An overview of topic modeling and its current applications in bioinformatics,» 2016.

[4] L. Hammoe, DETECCIÓN DE TÓPICOS Utilizando el Modelo LDA (trabajo final para especialista en ciencia de datos), BUENOS AIRES, 2018.

[5] L. Molina y J. Maldonado, «Implementación de un Método para la Clasificación Automática de Documentos Usando Tareas de Procesamiento de Lenguaje Natural y un Algoritmo de Máxima Entropía,» vol. 4, nº 1, pp. 1-9, 2017.

[6] J. Allan, J. Carbonel, G. Doddingto, J. Yamro y Y. Yan, «Topic Detection and Tracking Pilot Study Final Report,» 1998.

[7] A. Fernández Anta, P. Morere, L. Núñez Chiroque y A. Santos, «Sentiment Analysis and Topic Detection of Spanish *Tweets*: A Comparative Study of NLP Techniques,» vol. Procesamiento del Lenguaje Natural, nº 50, pp. 45-52, 2013.

[8] H. Gómez-Adorno, I. Markov, G. Sidorov, J.-P. Posadas-Durán, M. A. Sanchez-Perez y L. Chanona-Hernandez, «Improving Feature Representation Based on a Neural Network for Author Profiling in Social Media Texts,» nº 1638936, p. 13, 2016.

[9] F. Chiu Hsieh, R. F. Sandroni Dias y I. Paraboni, «Author Profiling from Facebook Corpora,» 11 th International Conference on Language Resources and Evaluation (LREC-2018), Miyazaki, Japan, 2018.

[10] P. Rosso, F. Rangel, I. Hernández Farías, L. Cagnina, W. Zaghouani y A. Charfi, «A survey on author profiling, deception, and irony detection for the Arabic language,» *2018,* vol. 12, nº e12275., 2018.

[11] R. Mishra, P. Prakhar Sinha, R. Sawhney, D. Mahata, P. Mathur y R. Ratn Shah, «SNAP-BATNET: Cascading Author Profiling and Social Network Graphs for Suicide Ideation Detection on Social Media,» de *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, Minneapolis, Minnesota, 2019.

[12] J. A. González Barba, «Aprendizaje profundo para el procesamiento del lenguaje natural (tesis de master),» *Universidad politécnica de Valencia,* 2017.

[13] Z. S. Harris, «Distributional structure 10(2-3):146–162,» 1954.

[14] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,» 24 Mayo 2019.

[15] K. Clark, M.-T. Luong, Q. V. Le y C. D. Manning, «ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS,» 23 Marzo 2020.

[16] P. Bojanowski, E. Grave, A. Joulin y T. Mikolov, Enriching Word Vectors with Subword Information, 2016.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado y J. Dean, Distributed representations of words and phrases and their compositionality, 2013.

[18] T. Mikolov, K. Chen, G. Corrado y J. Dean, Efficient estimation of word representations in vector space, 2013.

[19] F. Godin, B. Vandersmissen, W. D. Neve y R. V. d. Walle, «Multimedia lab @ ACL WNUT NER shared task: Named entity recognition for *Twitter* microposts using distributed word representations,» de *Proceedings of the Workshop on Noisy User-generated Text*, Beijing, China, Association for Computational Linguistics, 2015, pp. 146-153.

[20] J.-Á. González, F. Pla y L.-F. Hurtado, «ELiRF-UPV at TASS 2018: Sentiment Analysis in *Twitter* based on Deep Learning,» de *Proceedings of TASS 2018: Workshop on Semantic Analysis at SEPLN, TASS@SEPLN 2018, co-located with 34nd SEPLN Conference (SEPLN 2018)*, Sevilla, Spain, 2018.

[21] G. v. Rossum, «Python,» 1991. [En línea]. Available: www.python.org.

[22] JetBrains, «PyCharm,» 3 Febrero 2010. [En línea]. Available: www.jetbrains.com/pycharm/.

[23] «Tweepy,» [En línea]. Available: https://www.tweepy.org/.

[24] «PyMongo,» [En línea]. Available: https://pymongo.readthedocs.io/en/stable/.

[25] «Pickle,» [En línea]. Available: https://docs.python.org/3/library/pickle.html.

[26] R. Řehůřek, «Gensim,» Řehůřek, Radim, 2009. [En línea]. Available: https://radimrehurek.com/gensim/.

[27] D. Cournapeau, «scikit learn,» Junio 2007. [En línea]. Available: https://scikit-learn.org/.

[28] T. Oliphant, «Numpy,» 2006. [En línea]. Available: https://numpy.org/.

[29] «tkinter,» [En línea]. Available: https://docs.python.org/3/library/tkinter.html.

[30] «MongoDB,» 2007. [En línea]. Available: https://www.mongodb.com/.

[31] T. Preston-Werner, C. Wanstrath, P. J. H. y S. Chacon, «GitHub,» 8 Febrero 2008. [En línea]. Available: https://github.com/.

[32] «Ranks NL,» [En línea]. Available: https://www.ranks.nl/stopwords/spanish. [Último acceso: Enero 2021].

[33] A. Savand, «Github,» Mayo 2014. [En línea]. Available: https://github.com/Alir3z4/stop-words/blob/master/spanish.txt. [Último acceso: Enero 2021].

[34] «COUNTWORDSFREE,» Enero 2021. [En línea]. Available: https://countwordsfree.com/stopwords/spanish.

Glosario

- **Tesauro:** Una lista de palabras o términos controlados, empleados para representar conceptos
- **Token:** componente léxico con significado, generalmente obtenido de un texto u oración mediante diferentes técnicas para su posterior procesamiento.
- **Tweet:** Mensaje de máximo 280 caracteres escrito en una cuenta de *Twitter*. Es la característica principal de *Twitter* como plataforma.
- **Cuenta y usuario:** Hace referencia a un miembro de la plataforma *Twitter*. Se pueden buscar y referencia en la plataforma buscando su nombre precedido por una arroba "@".
- **Retweet:** Acción de compartir un *tweet* ya escrito, de otro usuario o propio, en una cuenta.
- **Hashtag:** Termino que define y caracteriza a un *tweet,* van precedidos de un "#", y son seguidos por cualquier palabra o conjunto de palabras, pueden servir para identificar a que tema hace referencia el *tweet.*
- **Interfaz:** Elemento de una aplicación, que permite a un usuario interaccionar con la misma de forma visual y clara, también denominada como GUI, Graphic user interface.
- **Json:** Formato especifico de ficheros, usualmente representado mediante pares del estilo: <Clave>:<Valor>

Apéndice 1: Cuentas de Twitter

Las siguientes tablas presentan las cuentas usadas para la creación del modelo y la cantidad de *tweets* recuperados en cada una

COCINA		DEPORTE	
@RecetasdeCocina	2358	@DeportesCuatro	3053
@Cocina_Monstruo	3086	@ABCDeportes	3126
@derechupete	2966	@2010MisterChip	3138
@mcalabajio	2483	@mundodeportivo	2932
@cocinaparados	1121	@diarioas	2330
@Dulcesbocados	2801	@CaracolDeportes	2475
@CocinayVino	3155	@valenciacf	2953
@CanalCocina	2933	@marca	2341
@directopaladar	3216	@FOXDeportes	3199
@RecetasdeCocina	2358	@ESPNtenis	3180
@GastroSER	2309	@MarcaBasket	3218
@GASTROactitud	2941	@EFEyDeporte	2422
@Gastronomiaycia	3075	@deportegob	2491
@DGastronomia	3219	@rfef	3098
@GastronomiaCyL	310	@Tokyo2020es	876
@VinosdeCebreros	1134	@teledeporte	2396
@Degusta_cyl	1701	@deportes_rtve	2502
@tierradesabor	2946	@podium_EE	3224
@MiquelSen	2877	@atletismoRFEA	2788
@GastronomadeGa2	1149	@Antena3Deportes	2967
@vlcgastronomica	2877	@mundodeportivo	2932
@andgastrotur	1790	@yosoynoticia_	3243
@GastronomiaVeg	1145	@sportsmadeinusa	2412
@gastrosingular	2620	@20mDeportes	3218
@HeraldoGastro	2985	@ESPNDeportes	3111
@gourmeturbano	3135	@atletismoSomos	2348
@bienmesabe_	3156	@EPdeportes	3156
@gourmetjournal	3120	@EsFutbol_Ingles	2271
@recetasen140	2939	@deporte_mujer	2935
@gdegastronomia	2357	@destelladeporte	427

TECNOLOGÍA E INFORMÁTICA		POLÍTICA	
@360_Hardware	3085	@PODEMOS	1612
@QNAPEspana	1442	@PSOE	2463
@elpais_tec	3195	@populares	1663
@xataka	3191	@CiudadanosCs	1026
@genbeta	3181	@InesArrimadas	1685
@MuyComputerPRO	3107	@sanchezcastejon	1958
@computerhoy	3147	@vox_es	1038
@juanklore	2622	@Santi_ABASCAL	733
@TecnosferaET	2885	@PabloIglesias	2080
@TecnoEspectador	3182	@Politica_LR	2960
@LaMMordida	3121	@ierrejon	2829
@Ftv_Fractal	2955	@tuvozalmundo	2468
@TopesdGama	3190	@juanpalop	1522
@Revista_ByteTI	2563	@AquiEuropa	3172
@elpais_tec	3195	@begonavillacis	1315
@pixeltech	3067	@Tonicanto1	1512
@jlacort	2996	@alvaro7carvajal	2061
@adslzone	3156	@lugaricano	2040
@ZoomNet_tve	2559	@Albert_Rivera	2228
@ReformaGadgets	2915	@IzquierdaUnida	899
@ENTERCO	2826	@MonederoJC	2509
@tecnocat01	2934	@ccifuentes	2894
@Teknautas	3235	@MisspoliticaMg	3140
@mixx_io	2746	@mundo_mas_justo	2772
@jmatuk	2727	@edosmilaragon	3178
@Giz_Tab	3063	@amnistiaespana	2648
@iSenaCode	3148	@OxfamIntermon	2729
@pisapapeles	3091	@Declaracion	2486
@MovilZona	3164	@reformainter	2983
@MalditaTech	698	@proceso	3110

VIDEOJUEGOS		OTROS	
@MeriStation	3152	@diezminutos_es	3237
@vidaextra	3240	@Felipez360	3094
@Eurogamer_es	3246	@Conmicas	1056
@3djuegos	3217	@skereunpesado	2515
@Videojuegos	3161	@SandraFerrerV	2567
@hobby_consolas	3196	@myh_tv	2828
@PSPlusES	2335	@AnimeMangaSpain	2321
@PokemonGOespana	3231	@LocaportuRopa	1175
@BRCDEvg	2220	@volvemos_	2444
@LVPesLoL	2979	@FundacionMigrar	2021
@TwitchES	432	@MigrarEsCultura	3207
@Xbox_Spain	2808	@LadyLauryCandy	1691
@Capcom_Es	1115	@perezreverte	2499
@KochMedia_es	1458	@TruthSeekerEs	1335
@FortniteEsp_	3048	@ENM_UNAM	3205
@marcaesports	3093	@CanaldeHistoria	3016
@LVPes	2539	@medicinisse	3184
@Warcraft_ES	2342	@FacMedicinaUNAM	3006
@Minecraft_ESP	1609	@PSICOLOGO	1028
@IGN_es	3199	@chincheto77	3102
@ESLspain	2109	@paulagonu	1281
@Esportmaniacos	3037	@javviercalvo	2040
@AnaitGames	2825	@lolaindigomusic	2741
@VideojuegosGAME	3142	@juanrallo	3122
@PlayStationES	2395	@_Cinefilos_	2926
@HPTXVideojuegos	3149	@FICM	1504
@comunidadxbox	3210	@edcerbero	2007
@Xbox_Jugones	2779	@nippon_es	3221
@ttdvideojuegos	2821	@culturainquieta	2804
@FolagoR	3058	@Rafael_delRosal	1686

Las siguientes tablas presentan las cuentas usadas para la comprobación de calidad del modelo y la cantidad de *tweets* recuperados en cada una

COCINA		DEPORTE	
@thermorecetas	3209	@ElGolazoDeGol	2704
@Gastro7islas	2888	@DAZN_ES	3108
@ComidasDelPeru	2561	@EFEdeportes	3182
@HortoGourmet	2663	@revistalideras	2673
@gastro_bloguers	2808	@COE_es	2808
@gastronomia_zgz	1568	@juegosolimpicos	2779
@DeMayorCocinero	2719	@Mercado_Ingles	2838
@chefghgonzalez	1017	@AS_amaliafra	2216
@RAGinforma	846	@TUDNUSA	2765
@gastronomia593	981	@NBAspain	2374
@delascosasdelco	887	@deportesyahoo	934
@igastronomia	3248	@mundosportextra	3048

TECNOLOGÍA E INFORMÁTICA		POLÍTICA	
@E1Am1g01nf0rma1	2768	@LPGPolitica	3214
@abc_tecnologia	3005	@pablocasado_	1981
@HardwareSfera	3171	@MasPais_Es	1067
@muycomputer	3207	@euroefe	3194
@ExpansionTecno	2153	@esglobal_org	2999
@muylinux	2834	@IreneMontero	1549
@TecnonautaTV	3025	@susanadiaz	2634
@_VanCajun	2962	@marianorajoy	1756
@unocero	3222	@UPYD	736
@htcmania	3222	@CsCValenciana	839
@daliadepaz	2506	@PoliticaPMA	2193
@Wikichava	2967	@oxfam_es	2422

VIDEOJUEGOS		OTROS	
@MeridiemGames	2355	@twin_melody	2045
@Nintenderos	3202	@IgnatiusFarray	1295
@VandalOnline	3212	@HistoriaDiceQue	2514
@GamereactorES	3161	@NoSoyLaGente	2458
@Ubisoft_Spain	2771	@eduardogavin	1948
@bethesda_ESP	3078	@Trabalibros	2722
@AlfaBetaJuega	3234	@relibertad	3221
@Videojuegos40	1121	@LaKefa	1110
@CallofDutyES	1745	@ecartelera	2812
@GeneracionXbox	3100	@viajar	3202
@EspVideojuegos	2740	@vickicastillo__	2971
@atomix	3238	@ProfesorSnape	2680