Generador de un conjunto de datos de entrenamiento de un sistema inteligente para predicción de género a partir del contenido de mensajes públicos de Twitter

Training dataset generation from public messages in Twitter for a gender prediction intelligent system

Pedro Llamas Do Espíritu Santo

Trabajo de fin de grado del Grado en Ingeniería Informática,
Facultad de Informática, Universidad Complutense de Madrid

27 de enero de 2021

Directora: Sara Román Navarro

Índice

Página

Resumen ..... **III**

Palabras clave ..... **III**

Abstract ..... **IV**

Keywords ..... **IV**

Capítulo 1. Introducción **1**

Capítulo 2. Objetivos **3**

Capítulo 3. Estado del arte **5**

Capítulo 4. Planificación del trabajo **8**

Capítulo 5. Tecnologías y herramientas **9**

5.1 Java 9

5.2 JavaFX 9

5.3 Twitter API 10

5.4 Twitter4j 11

5.5 R 11

5.6 Máquinas de Vectores de Soporte (SVM) 12

5.7 Intellij IDEA 13

5.8 R Studio 14

5.9 GitHub 15

Capítulo 7. Metodología e implementación **17**

7.1 Funcionamiento general 17

7.1.1 Fase de recolección de datos 17

7.1.2 Fase de procesamiento de los datos 20

7.1.3 Fase de validación mediante clasificador SVM 22

7.2 Componentes 22

7.2.1 Aplicación para generar el dataset 22

7.2.2 Script R 32

Capítulo 8. Resultados 35

Capítulo 9. Conclusiones y trabajo futuro 39

Apéndices 40

Características utilizadas 40

Palabras o caracteres en otros idiomas 44

Bibliografía 45

Anexos 48

Manual de uso 48

Twitter API 48

Recolección de usuarios 49

Recolección de mensajes (tuits) 51

Procesamiento del texto y extracción de características 52

Resumen

En este trabajo se ha desarrollado una aplicación generadora de conjuntos de datos de entrenamiento (dataset) a partir del texto contenido en los mensajes de un usuario de Twitter, con la finalidad de predecir, mediante el uso de sistemas inteligentes, el género de la persona detrás de un usuario en esa plataforma.

Tras obtener el acceso a la información de la plataforma utilizando las APIs de la misma, se identificó con un proceso automático una cantidad igual de usuarios hombres y mujeres cuyos mensajes estuvieran escritos en español además de otras características; y posteriormente se recolectaron los mensajes de los mismos.

A partir de estos mensajes se extrajeron y cuantificaron una serie de características lingüísticas, las mismas para cada usuario.

Finalmente, a partir de las características cuantificadas, se creó el correspondiente conjunto de datos con la información debidamente diferenciada por género.

Para llegar a la conclusión de que el conjunto de datos generado sirve para entrenar sistemas inteligentes, se validó con un clasificador basado en Máquinas de Vectores de Soporte, entrenándolo con dicho conjunto y comparando los resultados con los de estudios previos similares.

Palabras clave:

Twitter, Género, Sistema inteligente, Máquina de Vectores de Soporte, Entrenamiento, Conjunto de datos.

Abstract

In this work, an application has been developed that generates training data sets (dataset) from the text contained in the messages of a Twitter user, in order to predict, through the use of intelligent systems, the gender of the person behind a user on that platform.

After gaining access to the information on the platform using its APIs, an equal number of male and female users were identified with an automatic process whose messages were written in Spanish in addition to other characteristics; and later their messages were collected.

From these messages, a set of linguistic characteristics were extracted and quantified, the same for each user.

Finally, based on the quantified characteristics, the corresponding data set was created with the information duly differentiated by gender.

To reach the conclusion that the generated data set is valid for training intelligent systems, it was validated with a classifier based on Support Vector Machines, training it with the generated set and comparing the results with those of previous similar studies.

Keywords:

Twitter, Gender, Intelligent *System, Support Vector Machine, Training, Dataset.*

Capítulo 1. Introducción

Las personas han estado compartiendo información y textos escritos por ellos mismos desde los inicios de Internet, así lo hacían las pocas personas que en esos momentos tenían acceso, enviando correos electrónicos o incluso publicando mensajes o algún documento en los pequeños foros que se creaban en aquellos tiempos. Pero con el auge de las tecnologías y el desarrollo de internet, además de su masificación y llegada a gran parte de la población han aparecido nuevos medios de comunicación, sobre todo destinados a la interacción de las personas, como son las redes sociales.

En estas redes sociales es donde hoy en día se produce el mayor tráfico de información a nivel global, y al estar destinadas a la interacción de sus usuarios, en ellas las personas publican sus mensajes; ya sea manteniendo una comunicación con otras o simplemente contando algún suceso propio o haciendo alguna reivindicación. Ejemplos como Facebook o Twitter son muy claros, redes sociales destinadas principalmente a la interacción entre sus usuarios, de las que gran parte se realizan en formato de texto y en las que cada persona publica normalmente sus propios mensajes.

Estos mensajes contienen información de gran valor, si lo vemos desde el punto de vista de las empresas, ellas se dedican a analizar esta información creando perfiles de los usuarios, adaptando así su publicidad y negocio a estos perfiles y aumentando su nivel de éxito entre la población objetivo a quien dirigen sus productos o servicios.

Viendo que es posible analizar a un usuario para saber qué es lo que más demanda o lo que más le puede interesar a partir únicamente de la información que deja en las redes sociales, no es diferente de estas técnicas el analizar esa misma información para diferenciar el género del mismo, algo muy útil para realizar estudios de género o incluso facilitar la identificación de personas que cometen delitos de odio.

Es aquí donde toma gran valor el texto que ese usuario ha publicado en la red social.

En la escritura cada persona plasma sus rasgos sobre el texto haciéndolo “único”, por tanto, a un autor se le puede identificar por sus textos. Esto es algo que se ha utilizado ya, por ejemplo, para la atribución de la autoría de una obra a un determinado escritor o en sistemas de detección de plagios. Así, si una persona puede ser identificable lo mismo puede aplicarse a un grupo concreto de la población, como es el caso de los dos grupos diferenciados de hombres y mujeres.

Este análisis del texto está basado en la estilometría [1] (cada texto tiene ciertas características y marcadores propios de su autor) y es la clave en el desarrollo de este trabajo.

Capítulo 2. Objetivos

El objetivo final de este trabajo es conseguir crear una herramienta que genere un conjunto de datos (dataset) válido para entrenar un sistema inteligente y conseguir que el mismo pueda diferenciar el género de la persona detrás de una cuenta de Twitter a partir del texto de sus tuits públicos.

El trabajo se ha centrado en la parte de generación del dataset, ya que recolectar y preparar los datos requiere de un mayor esfuerzo que en última instancia, entrenar un sistema inteligente. Si no se realiza una buena elección de las variables a observar y se consigue una buena muestra de datos, el sistema por muy eficiente que sea no conseguirá unos resultados aceptables [2].

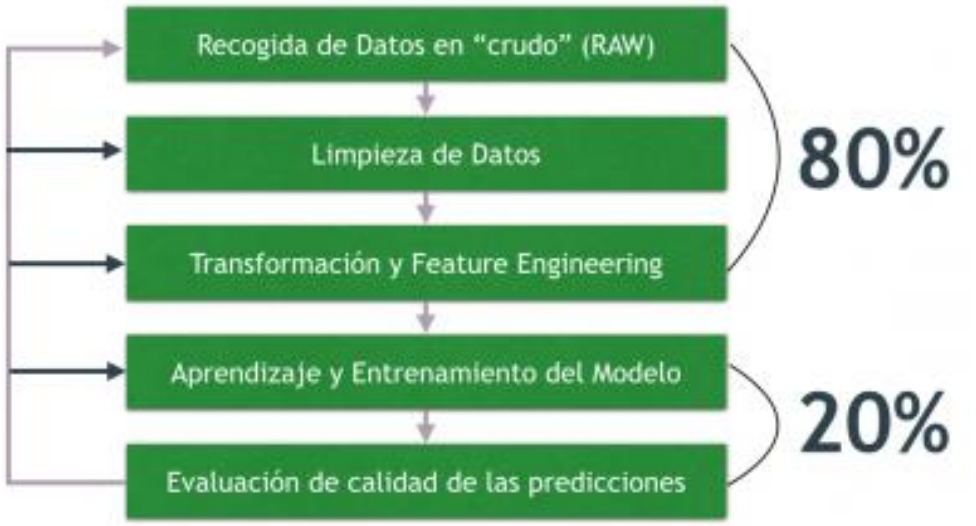


Figura 2.1. División del tiempo dedicado en la creación de un producto de aprendizaje automático

"Los algoritmos sólo pueden extraer sentido de los datos que le proporcionamos, no tienen capacidad de intuición como los humanos, sea esto bueno o malo, por lo que el éxito del sistema depende principalmente de los datos de entrada" [2]

Tomando como base el artículo *Author gender identification from text* [3], del que se hablará más adelante, se han elegido ciertas características del texto de los tuits para analizar (más en detalle en el punto 7.1.2 de este documento) y el algoritmo de Máquina de Vectores de Soporte (SVM), del cual también se habla más adelante, como algoritmo para comprobar la calidad del dataset, ya que es el que mejores resultados obtiene para una clasificación con unas variables similares.

Por ello, se busca obtener un dataset válido para entrenar cualquier sistema inteligente que clasifique a un usuario de Twitter según su género; y que con el mismo, al entrenar un sistema inteligente basado en SVM, se obtenga una precisión de clasificación similar a la que se obtiene en el artículo citado anteriormente; teniendo en cuenta que para este trabajo se han analizado menos características, el idioma del texto no es el mismo (ingles frente a español) y el contexto de los mensajes tampoco es el mismo (correos electrónicos privados frente a tuits públicos).

Capítulo 3. Estado del arte

Desde hace tiempo la identificación del género de una persona ya sea por su voz, sus rasgos faciales o por sus textos es objeto de estudio. Gracias a ello se puede identificar la autoría de una obra, afinar el filtro para la detección de personas que hayan cometido delitos en internet, facilitar una variable más de información a las empresas de cara a que generen su publicidad en torno a un grupo de clientes concreto...

Esto ha llevado a que se desarrollen estudios y programas para buscar las técnicas y métricas más eficientes que puedan realizar estas identificaciones mediante algoritmos de *Machine learning* [4].

Por ejemplo, las señales acústicas del habla, además del contenido lingüístico, pueden transmitir información sobre el género, el grupo social o incluso el tamaño de una persona. Si se transforma esta señal de voz en una secuencia de vectores de características basados en el espectro, puede analizarse para identificar esas características [5].

La identificación del género mediante rasgos faciales también ha tenido su aplicación. Gracias a la extracción de información sobre las características geométricas de una cara, el mapeo de expresiones faciales, etc. y con la ayuda de redes neuronales, por ejemplo, en una búsqueda en una base de datos se puede reducir en gran cantidad el tiempo de búsqueda al limitar la etapa de búsqueda posterior a una base de datos masculina o femenina [6].

En países de oriente medio se ha estudiado el uso de la identificación del género a través del texto de publicaciones en redes sociales como herramienta para identificar a los propietarios de identidades falsas con el fin de perseguir delitos [7].

En gran parte de los artículos esta identificación del género a partir de texto se ha realizado analizando textos en inglés. Un buen estudio; y sobre el que se ha basado este trabajo es *Author gender identification from text* [3].

En ese artículo se explican las bases de una aplicación para la identificación de género a partir de mensajes personales de correo electrónico. Los mensajes con los que trabajan son de una base de datos en internet (Enron Corpus) de los cuales finalmente se usaron 8970 (4947 de hombres y 4023 de mujeres) con un tamaño medio de 116 palabras.

Aunque el trabajo está desarrollado para textos en inglés, las características lingüísticas de los textos son extrapolables al español. Se han determinado así cinco categorías de estas características que están relacionadas con el género:

- Basadas en caracteres (número de espacios en blanco, mayúsculas, dígitos, etc.).
- Basadas en palabras (media del tamaño de palabra, métricas relacionadas con la riqueza del vocabulario, etc.).
- Sintácticas (número de comas, puntos, exclamaciones, interrogaciones, etc.).
- Basadas en la estructura (número de líneas, párrafos, oraciones, etc.).
- Basadas en palabras funcionales (número de artículos, pronombres, conjunciones, etc.).

Además de estas categorías también se han tenido en cuenta otras características vinculadas al género, como pueden ser el uso de adjetivos afectivos o palabras que denoten emociones.

Tras la representación numérica de estas características y su normalización (adaptar los valores para que todas estén comprendidas en unos ciertos rangos) se establece la eficacia de varias técnicas de clasificación, en concreto *Máquinas de Vectores de Soporte (SVM)*, *Regresión Logística Bayesiana* y Arboles de decisión (Algoritmo AdaBoost).

Los experimentos realizados usando esas tres técnicas concluyen que el mejor sistema para la identificación del género con esos datos es un clasificador SVM con kernel radial, que en determinadas circunstancias llega a obtener una precisión del 82,23%. El artículo también expone que a mayor tamaño de los mensajes mejor es la predicción de los diferentes algoritmos.

Fue gracias a la información aportada por todos estos estudios, en el caso de este trabajo, que se decidiera que los mensajes que se usarían para generar el dataset fueran extraídos de usuarios públicos en la plataforma Twitter, una red social basada en el microblogueo con un máximo de 280 caracteres por mensaje, que aporta una información muy similar en cuanto a contenido de los mismos al último artículo mencionado.

La técnica de clasificación elegida sería un SVM con kernel radial, ya que se probó que es el que mejor precisión arroja para un conjunto de datos similar.

Capítulo 4. Planificación del trabajo

El proyecto se comenzó con la división en diferentes fases del trabajo a realizar para poder completar el mismo.

Las fases que se identificaron fueron:

- Búsqueda de información sobre trabajos previos similares.
- Recolección de datos de Twitter.
- Procesamiento de los datos.
- Validación del dataset mediante un clasificador SVM.

Además del estudio y revisión de los artículos mencionados en apartados anteriores, se realizó una búsqueda previa de proyectos similares a través de buscadores como Google y más específicamente en la web https://github.com/, una plataforma con una gran cantidad de proyectos de código abierto.

No se encontró ningún trabajo similar relevante, ya que los más parecidos estaban creados para textos en inglés y el procesamiento y tratamiento de los datos no era aplicable a este proyecto.

Sin embargo, si se encontraron varias páginas que describen con claridad el proceso para la creación de un clasificador SVM en R, que resultaron muy útiles para el desarrollo de esa parte del trabajo. [8][9][10]

La fase de recolección de los datos y su procesamiento se decidió realizar en lenguaje Java, usando IntelliJ IDEA como entorno de desarrollo, por ser un entorno familiar y que permite el trabajo en este lenguaje. En cambio, la fase de validación del dataset se programó en lenguaje R y se usó R Studio como entorno para su desarrollo.

También se eligió GitHub como repositorio para almacenar el código y el resto del trabajo y tener una copia de seguridad versionada del mismo.

Estas fases de describen con más detalle en el capítulo 7 del documento, en el punto 7.1.

Capítulo 5. Tecnologías y herramientas

Para el correcto desarrollo de este trabajo he tenido que obtener conocimientos sobre diversas tecnologías y herramientas. De algunas de ellas ya tenía conocimientos más avanzados o básicos, otras he tenido que investigarlas más a fondo.

5.1 Java

Java [11] es un lenguaje de programación orientado a objetos, desarrollado originalmente por James Gosling, de Sun Microsystems y cuya sintaxis deriva en gran parte de los lenguajes *C* y *C++*. Las aplicaciones Java se compilan a bytecode, que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura del computador.

Es uno de los lenguajes más usados para programar en todo el mundo, por lo que tiene una extensa biblioteca de recursos que proporcionan un gran apoyo a la hora de desarrollar cualquier aplicación. Además, más específicamente, cuenta con la biblioteca *twitter4j* de la que la se habla más adelante.

Debido a estas características y a ser un lenguaje de programación con el que ya estaba bastante familiarizado, la parte de extracción y procesamiento de los datos de este trabajo se desarrolló en este lenguaje.

5.2 JavaFX

JavaFX [12] es una biblioteca de Java que permite a los desarrolladores, entre otras muchas funciones, diseñar interfaces gráficas. Se podría decir que el sucesor de la librería *Swing*.

Para construir la interfaz de una aplicación en JavaFX se utiliza *FXML*, un lenguaje de marcado declarativo basado en *XML*; aunque también se puede usar JavaFX *Scene Builder* para diseñarla interactivamente.

De nuevo, por familiaridad con esta biblioteca, se aplicó esta tecnología para diseñar la interfaz de la parte java del proyecto, implementando la misma con un fichero *.fxml*.

5.3 Twitter API

Desde hace varios años, Twitter cuenta con un conjunto de API (Interfaz de Programación de Aplicaciones) accesibles y que permiten crear software que pueda integrarse con la plataforma [13]. Estas API ofrecen funciones e información sobre las cuentas y usuarios, los mensajes directos, los anuncios y los tuits y sus respuestas. Existen tres API diferentes:

- *Streaming API*. Proporciona los tuits en tiempo real, es decir, permite usar funciones con tuits que se estén publicando en ese mismo instante o publicar tuits propios, entre otras cosas.
- *Search API*. Da acceso a tuits con una antigüedad de hasta siete días, pudiendo establecer filtros en la búsqueda como lenguaje, geolocalización…
- *REST API*. Con ella se puede acceder a todos los datos de Twitter además de poder realizar cualquier otra operación de la plataforma.

Gracias a la *Search API* y a la *REST API* se consigue toda la información necesaria para el desarrollo del proyecto, aunque son necesarias una serie de claves para verificar el acceso. Para obtenerlas hay seguir una serie de pasos de registro como desarrollador en la plataforma de Twitter, los cuales se explicarán con más detalle en el anexo del manual.

Por último, cabe destacar que hay diferentes tipos de acceso a estas API dependiendo del tipo de cuenta de desarrollador que se tenga en Twitter. La versión gratuita, que es la que se ha utilizado, tiene un límite de 450 peticiones cada 15 minutos para las búsquedas (búsqueda de usuarios) y 1500 peticiones cada 15 minutos para obtener el timeline de un usuario (extraer los tuits de un usuario).

5.4 Twitter4j

Twitter4j [14] es una biblioteca de código abierto, no oficial, para la API de Twitter.

Funciona en cualquier versión de Java igual o superior a Java 5 y es probablemente la biblioteca más usada en este lenguaje para el trabajo con la información que provee la plataforma. Permite una fácil integración con Java y cuenta con varios métodos para reproducir cualquier funcionalidad de Twitter.

Por ello he tomado esta librería como la mejor opción para el manejo de la API de Twitter desde Java.

5.5 R

R [15] es un lenguaje de programación interpretado de software libre que está enfocado al análisis estadístico. Es uno de los lenguajes más utilizados en investigación científica, sobre todo en los campos del aprendizaje automático, la minería de datos, etc. Su desarrollo actual es responsabilidad del R Development Core Team.

Proporciona una gran cantidad de herramientas estadísticas (modelos lineales y no lineales, test estadísticos, algoritmos de clasificación, etc.) y la posibilidad de generar gráficas, algo muy útil en este tipo de estudios.

En este lenguaje, el usuario no programa “tradicionalmente”, sino que se trata de un método de ensayo-error en el que cuando el usuario se equivoca, se vuelve a probar hasta obtener un resultado satisfactorio.

Se ha elegido esta tecnología ya que la ***biblioteca e1071***, la cual puede usarse en R, contiene implementaciones para varios métodos de aprendizaje estadístico, en concreto *SVM*, que es necesario para la segunda parte de este trabajo (validación del dataset). Además, ya contaba con unos conocimientos básicos del uso del mismo que refresqué con la búsqueda de información básica del lenguaje [8].

5.6 Máquinas de Vectores de Soporte (SVM)

Desarrollado en la década de los 90, las Support Vector Machines (SVM) son un tipo de algoritmo de maching learning aplicable a problemas de regresión y clasificación, aunque son comúnmente usadas como modelos de clasificación [9][10].

Están fundamentas en el *Maximal Margin Classiffier*, basado en el concepto a su vez de *Maximal Margin Hyperplane*.

En los artículos *Máquinas de Vector Soporte (Support Vector Machines, SVMs)* [9] y *MÁQUINAS DE VECTOR SOPORTE* [10] se describe clara y brevemente los conceptos y el marco teórico del SVM y el hiperplano.

Una breve descripción del mismo es que en un espacio euclídeo p-dimensional, el hiperplano es un subespacio plano, de dimensión p-1, que divide el espacio en dos mitades separando perfectamente las observaciones.

Normalmente la mayoría de las observaciones en un caso real no se pueden separar linealmente de forma perfecta (no puede obtenerse el *Maximal Margin Hyperplane*), por lo que se busca obtener un hiperplano que casi separe las clases (*Support Vector Classiffier*), permitiendo que algunas observaciones se encuentren en el lado incorrecto del margen. El ajuste de ese margen genera un problema de optimización.

Para esto es importante tener en cuenta el hiperparámetro de tuning C. Este parámetro controla el número y severidad de las posibles violaciones del margen (y del hiperplano) que se admiten en el ajuste del modelo. Cuando el valor de C es pequeño, el margen es más ancho, y más observaciones pueden violarlo (la cuales se convierten en vectores de soporte). Cuando C sea más grande, menos observaciones podrán violar ese margen, generándose menos vectores de soporte.

En la práctica este parámetro se ajusta mediante validación cruzada.

Se puede intuir por las definiciones anteriores, que el *Suppor Vector Classifier* obtiene buenos resultados cuanto más lineal sea el límite entre las observaciones. Un método para a aplicar cuando la separación de los grupos

de las observaciones sea no lineal consiste en aumentar las dimensiones del espacio original.

Es aquí donde entra en juego el método de *Máquinas Vector Soporte* (SVM). Tiene los mismos fundamentos que el *Suppor Vector Classifier, pero además* permite el uso de kernels (función que devuelve el resultado del dot product entre dos vectores realizado en un nuevo espacio dimensional distinto al espacio original [9]).

Gracias al estudio previo *Author gender identification from text* [3] se sabe que el kernel óptimo para el cálculo de las observaciones generadas con el análisis de las características lingüísticas de un texto es el kernel radial, el cual tiene una representación similar a la siguiente y es el que usaré en el SVM:

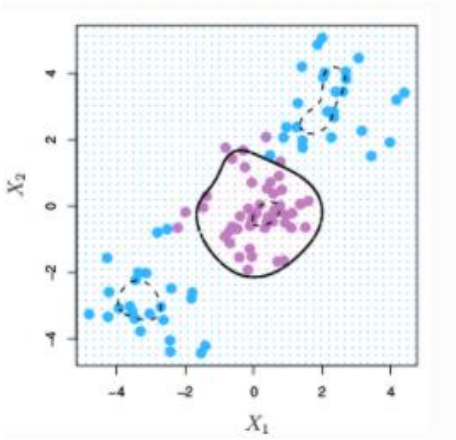


Figura 5.1. Imagen de SVM con un kernel radial

5.7 Intellij IDEA

Se trata de un entorno de desarrollo integrado (IDE) propiedad de JetBrains que sirve para el desarrollo de programas informáticos [16].

Soporta una gran cantidad de lenguajes de programación, entre ellos Java que es el que se usa mayoritariamente en el proyecto.

Además, se puede conectar directamente con GitHub, lo que hace que la carga de versiones del código en el repositorio sea más sencilla.

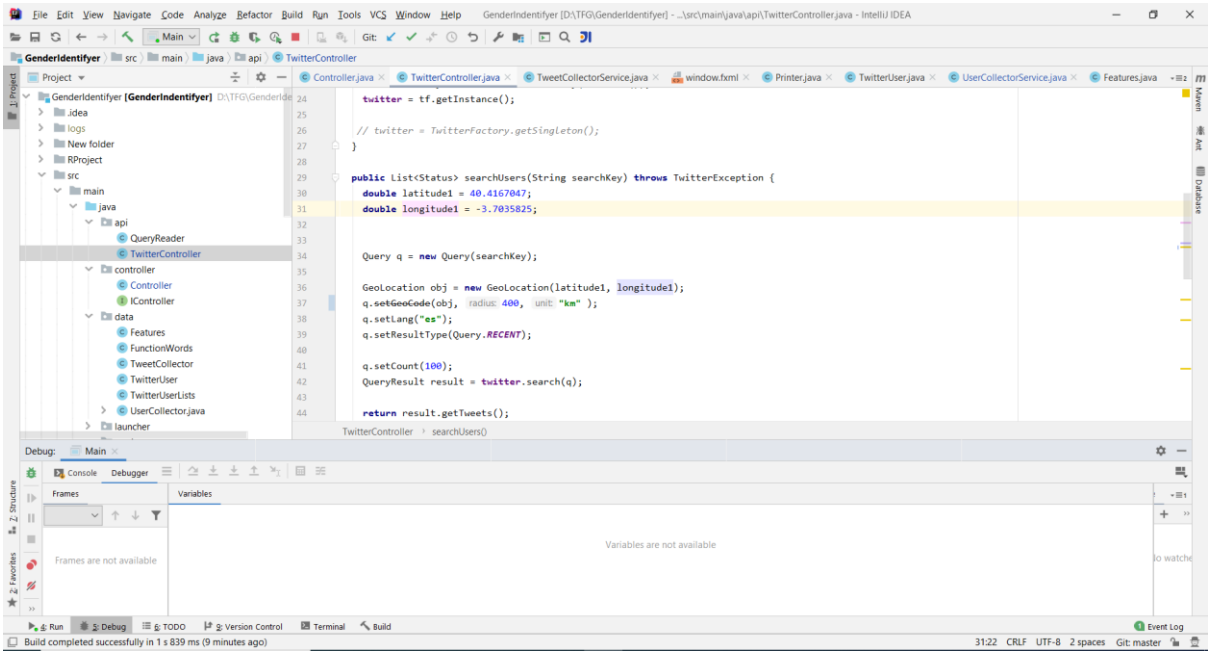


Figura 5.2. Vista del entorno de desarrollo Intelli IDEA

5.8 R Studio

Otro entorno de desarrollo integrado, en este caso para el lenguaje R y de software libre [17].

Cuenta con una consola, un editor de texto que admite la ejecución directa de código y diversas herramientas para depurar y gestionar el desarrollo en estos lenguajes. También cuenta con una ventana en la que ver archivos o gráficas creadas con los propios scripts.

La decisión de usar este entorno de nuevo fue el ya estar familiarizado mínimamente con él, además de ser uno de los más populares para el desarrollo en R.

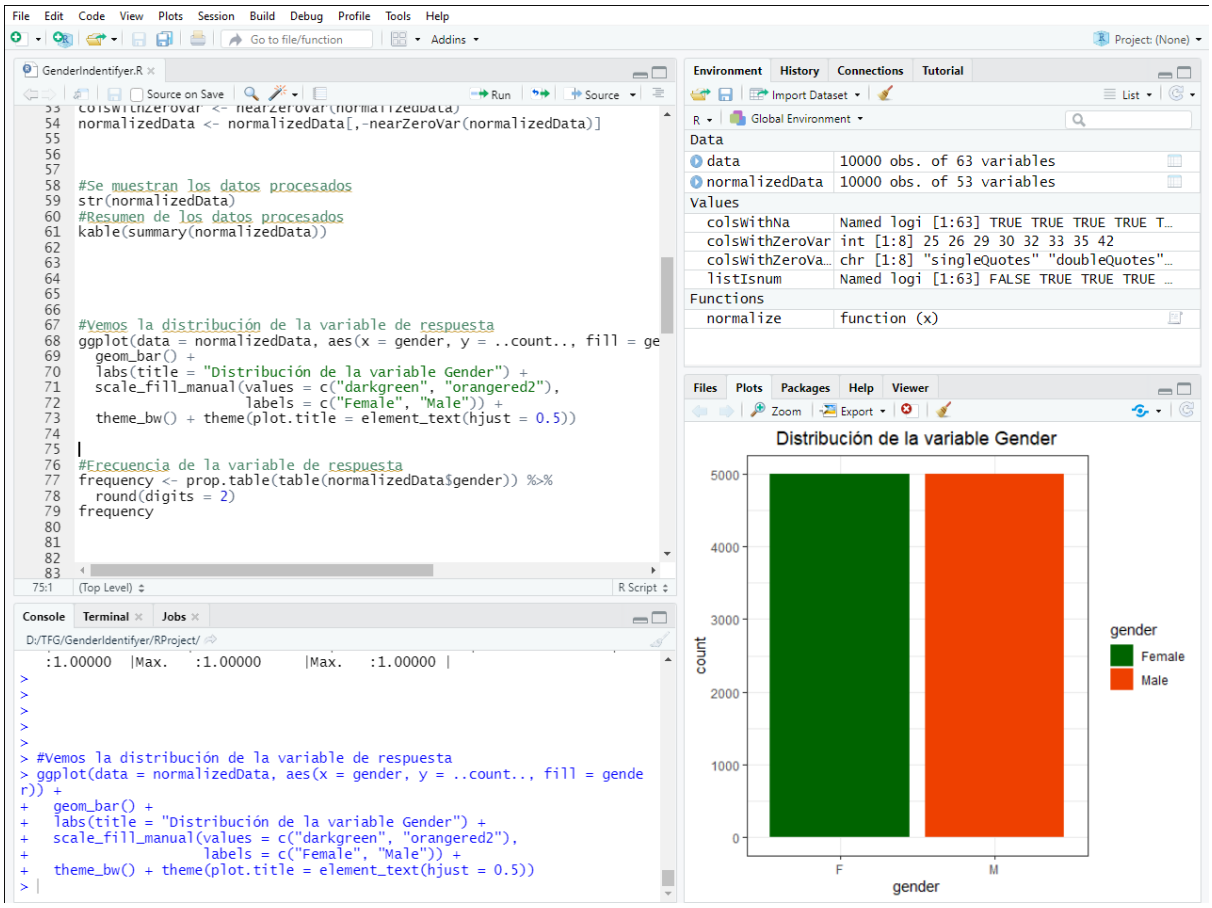


Figura 5.3. Vista del entorno de desarrollo R Studio

5.9 GitHub

Es una plataforma usada para el desarrollo colaborativo en la que principalmente se alojan proyectos mediante un sistema de control de versiones basado en Git.

Gracias a este control de versiones GitHub ofrece una gran versatilidad para la organización y gestión de las diferentes subidas (commits) del código de una aplicación.

Este es el motivo principal por el que se ha usado, además de así tener una copia de seguridad del trabajo en la nube para evitar cualquier problema que pudiera ocurrir en el equipo local.

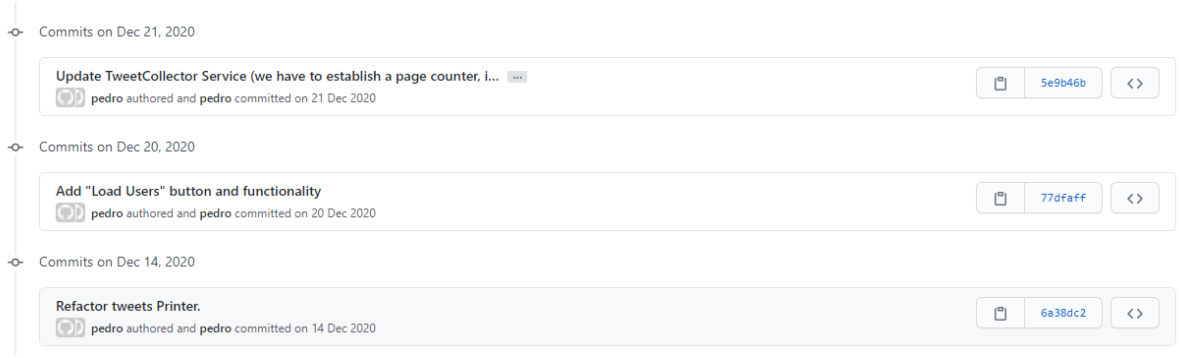


Figura 5.4. Historial de commits en la plataforma GitHub

Capítulo 7.

Metodología e implementación

7.1 Funcionamiento general

El desarrollo de este trabajo está inspirado en el artículo *Author gender identification from text* [3]. En él se explican las características en cuanto a estilometría que se deben buscar en un texto para identificar el género de su autor y como deben procesarse para posteriormente entrenar un clasificador basado en SVM.

Así, la funcionalidad de este proyecto se puede dividir en tres fases claramente diferenciadas, las cuales se describirán a continuación.

7.1.1 Fase de recolección de datos

En esta fase se extraen de Twitter usando su API todos los datos necesarios para generar posteriormente un dataset valido.

7.1.1.1 Recolección de usuarios

La recolección de los usuarios es el primer paso necesario para llegar al objetivo final.

Se realiza una búsqueda automática de palabras claves (comúnmente usadas y propias del español) en los tuits más recientes del timeline general de Twitter con ubicación en España. La finalidad es extraer el usuario que ha publicado ese tuit, el cual se guardará en un fichero para que posteriormente sea procesado.

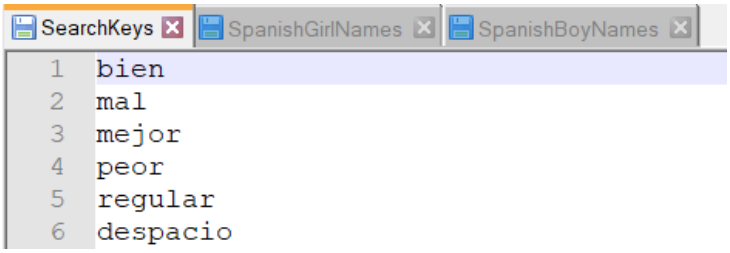


Figura 7.1. Archivo con las palabras claves en español para la busqueda

Para saber si el usuario es hombre o mujer y poder guardarlo, de manera automatizada, se compara el nombre que tiene en Twitter con una lista de nombres en español, en principio identificables exclusivamente con el género masculino o con el femenino.

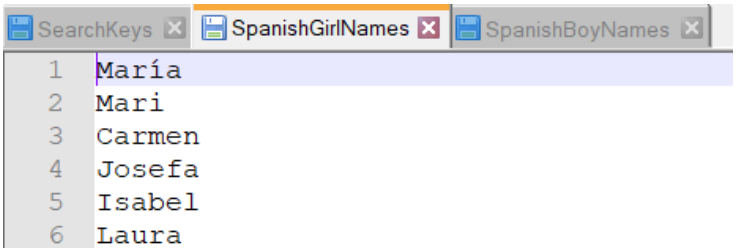


Figura 7.2. Archivo con los nombres de mujeres en español para comparar

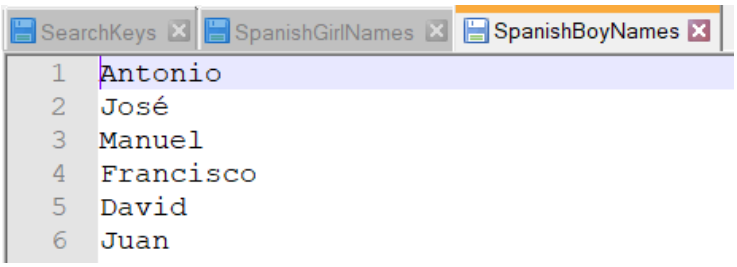


Figura 7.3. Archivo con los nombres de hombres en español para comparar

Además, se comprueba que el usuario no sea una cuenta con mucha repercusión para evitar que no sea la misma persona el usuario público que la persona real que escribe los tweets (usuarios famosos que le encarguen la gestión de sus redes sociales a otras personas).

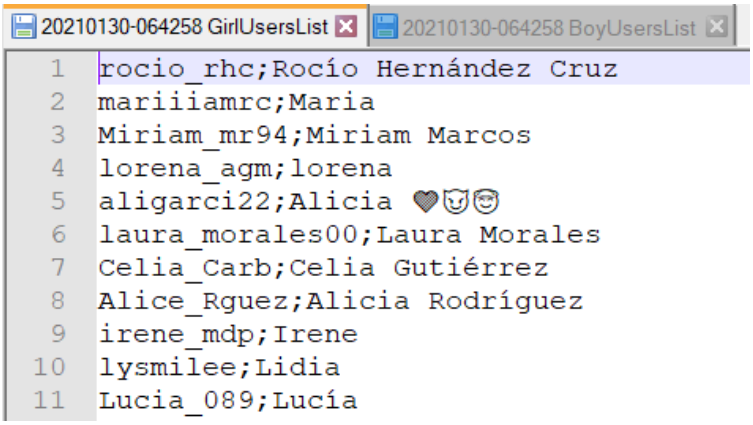


Figura 7.4. Archivo con los usuarios guardados de género femenino

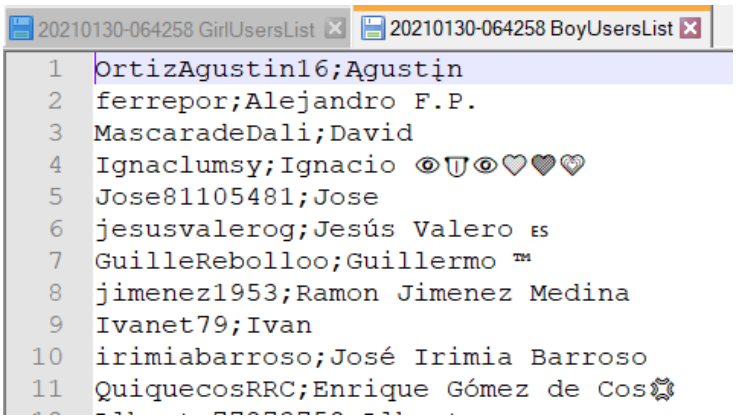


Figura 7.5. Archivo con los usuarios guardados de género masculino

Puede verse el proceso más detalladamente en el punto 7.2.1.3 del documento.

7.1.1.2 Recolección de tweets

Con los usuarios ya recolectados se realizan varias consultas por usuario a Twitter, la finalidad es obtener los últimos 100 tuits escritos por el usuario.

De forma automática, se comprueba que no sea una cuenta comercial o dedicada a la publicidad, cuyo lenguaje podría estar condicionado; y mediante una lista de palabras y caracteres de otros idiomas (apéndice I), que todos los tuits estén escritos es español.

Si finalmente el usuario cumple las condiciones, los tweets se guardan en el archivo correspondiente.

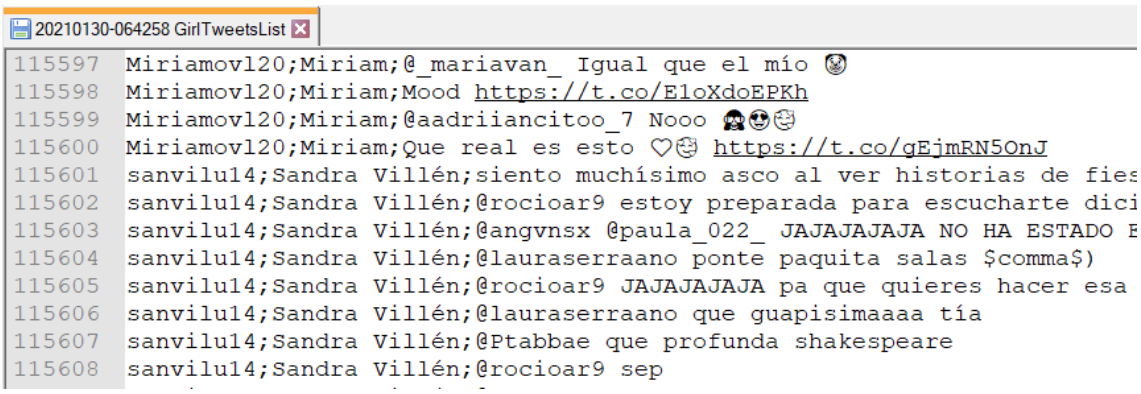


Figura 7.6. Archivo con los tuits de usuarios guardados de género femenino

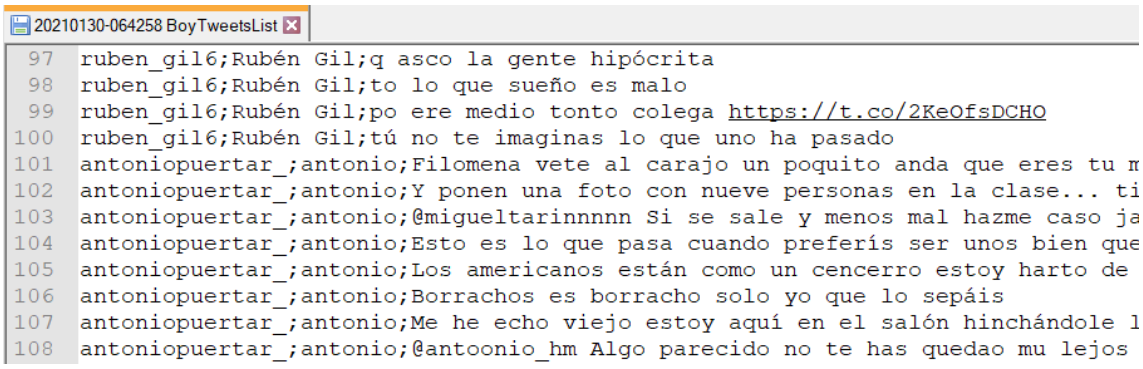


Figura 7.7. Archivo con los tuits de usuarios guardados de género masculino

Puede verse el proceso más detalladamente en el punto 7.2.1.4 del documento.

7.1.2 Fase de procesamiento de los datos

Una vez finalizada la fase de recolección de datos, se pasa a procesar todos los tuits de cada usuario. Para cada usuario se establecen una serie de características que se van cuantificando a medida que se recorren los tuits. Estas características se pueden agrupar en 7 grupos que son:

- Características basadas en caracteres (detallado en el apéndice A).
- Características propias de Twitter (detallado en el apéndice B).
- Características basadas en palabras (detallado en el apéndice C).
- Características métricas o medidas sobre las palabras (detallado en el apéndice D).
- Características basadas en palabras sintácticas (detallado en el apéndice E).
- Características basadas en palabras funcionales (detallado en el apéndice F).
- Características basadas en palabras psicolingüísticas (detallado en el apéndice G).
- Características basadas en palabras vinculadas al género (detallado en el apéndice H).

A diferencia del artículo *Author gender identification from text* [3] se han descartado las características que hacen referencia a enunciados, oraciones, etc., ya que al tratarse de mensajes breves no tiene sentido ni es posible encontrar estas estructuras, aunque sí que se han cuantificado los saltos de línea.

Tampoco se han incluido los verbos auxiliares ni el artículo “lo” ya que también puede ser un pronombre personal y de momento se ha optado por no cuantificarlos.

También se han descartado las “pro-sentences” (palabra o expresión funcional que sustituye a una oración completa cuyo contenido se puede recuperar del contexto [18]) ya que al no diferenciar estructuras como las oraciones son difíciles de determinar; y las palabras específicas de género.

Otras características como la frecuencia de distribución de las palabras o las características LIWC [19] se han omitido por la dificultad de su implementación.

Una vez procesados todos los usuarios, se guarda el dataset final con todas las características debidamente cuantificadas en un fichero. En la primera línea se encuentra el nombre de cada una de las características separadas por un espacio y por cada línea siguiente, se encuentra la información de un usuario, identificado únicamente por el género (*M* hombre y *F* mujer) y seguido por el resto de los valores de las características, todas ellas separadas por un espacio en blanco y siempre en el mismo orden.

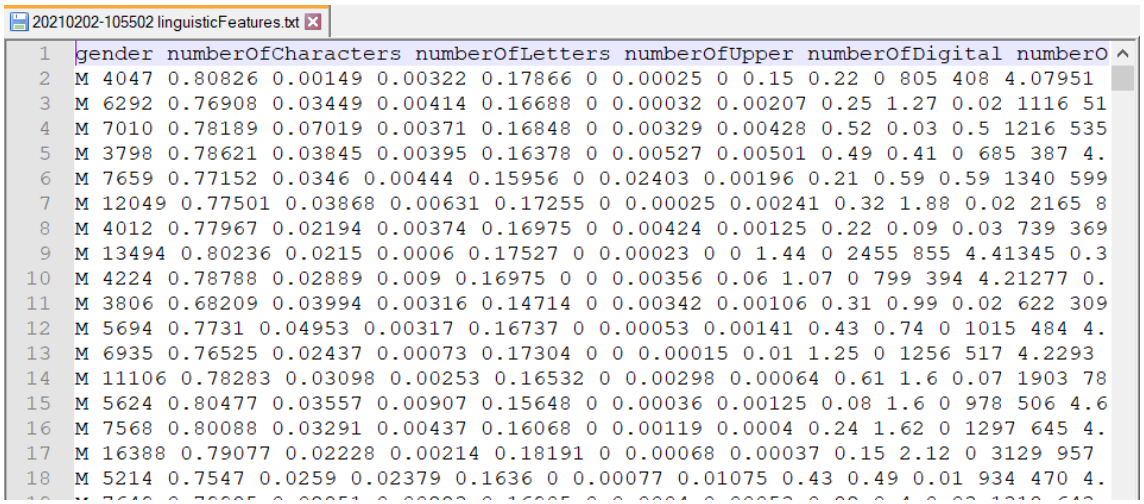


Figura 7.8. Archivo que contiene el conjunto de datos final

7.1.3 Fase de validación mediante clasificador SVM

Se trata de la fase final del trabajo. Con el dataset ya creado se procede a realizar un entrenamiento de un clasificador basado en *Máquinas de Vectores de Soporte* (SVM) con el fin de concluir si el dataset es válido para el entrenamiento de sistemas inteligentes.

Esta fase, en este trabajo, se utiliza como una fase puramente de test o validación, ya que gracias a ella se verifica que el dataset generado por la aplicación cumple con unos mínimos de precisión y fiabilidad para el entrenamiento de este tipo de sistemas.

De esta manera, basándonos en los resultados obtenidos en *Author gender identification from text* [3], se considera que una precisión igual o superior al 75% sería un buen resultado y por tanto el dataset sería válido, teniendo en cuenta que en este proyecto se han analizado un número menor de características.

7.2 Componentes

Este trabajo consta de dos componentes claramente diferenciados. Por un lado, la aplicación encargada de extraer y procesar los datos de Twitter generando finalmente un dataset; y por otro el script en R que se encarga de entrenar un clasificador SVM con el que verificar la precisión del dataset generado.

7.2.1 Aplicación para generar el dataset

La aplicación principal está desarrollada en Java, ya que ofrece una gran variedad de alternativas gracias a la gran cantidad de código abierto que existe.

Está dirigida por una clase controlador llamada *Controller.java*

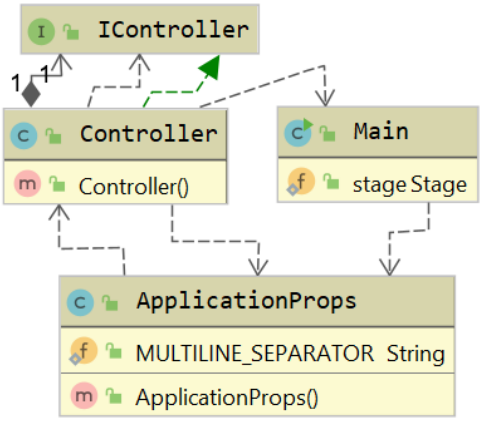


Figura 7.9. Relaciones clase Controller y ApplicationProps

Gestiona las barras de progreso, los diálogos y todas las acciones de la aplicación. La interfaz se genera con *JavaFX*, estando así manejada por un archivo *.fxml* (*window.fxml*) y simplificando la parte del diseño de la vista del proyecto.

Es importante destacar la carpeta *properties* que se encuentra en el mismo directorio que el proyecto o ejecutable.

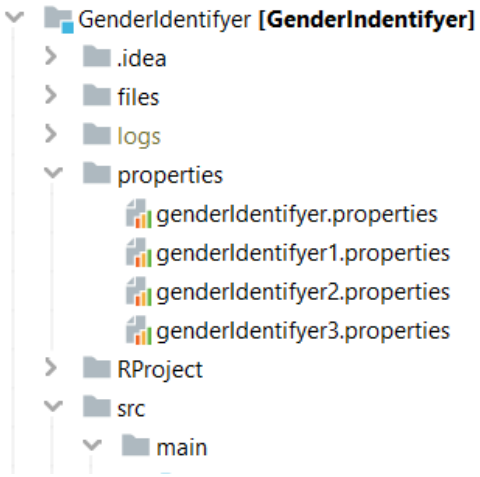


Figura 7.10. Estructura de los ficheros de propiedades en el proyecto java

Es necesario que exista, así como los archivos que contiene.

El archivo *genderIndentifyer.properties* almacena los valores de los cuadros de texto de la aplicación para que estos se mantengan cada vez que se inicia la aplicación. Además, debe contener las claves necesarias para el acceso a las APIs de Twitter.

genderIdentifyer.properties

```
1
2  boyNamesText=D:\TFG\GenderIdentifyer\files\SpanishBoyNames
3  girlNamesText=D:\TFG\GenderIdentifyer\files\SpanishGirlNames
4  searchKeysText=D:\TFG\GenderIdentifyer\files\SearchKeys
5  continueBoyUsersText=D:\TFG\GenderIdentifyer\20210130-064258 BoyUsersList
6  continueGirlUsersText=D:\TFG\GenderIdentifyer\20210130-064258 GirlUsersList
7  maxUsersText=5000
8  continueBoyTweetsText=D:\TFG\GenderIdentifyer\20210130-064258 BoyTweetsList
9  continueGirlTweetsText=D:\TFG\GenderIdentifyer\20210130-064258 GirlTweetsList
10 consumerKey=Q4s2bIwpEEFcYT4zOtjgWlNOM
11 consumerSecret=2YN2Mjl662s6HYSJhaYrkvBpVMppM5rHiWWeAsSQZabT0bCYbM
12 accessToken=1238876335789015042-fxtakDFzXhFl0XC8cAK6qrKdMNQSDn
13 accessTokenSecret=UiUMPukyOccDWd5oz21ccinZTgQuQVWWk4O9HTAnIlxzI
```

Figura 7.11. Contenido del archivo de propiedades principal

Los archivos *genderIndentifyer.properties1, genderIndentifyer.properties2, genderIndetifyer.properties3* guardan únicamente las claves para el acceso a las APIs de Twitter en el mismo formato que el anterior, pero está vez habrá tres conjuntos de claves diferentes (uno por fichero). Gracias a ello se pueden obtener los tuits de los usuarios rotando las claves cuando estas llegan al máximo de peticiones.

genderIdentifyer1.properties | genderIdentifyer2.properties | genderIdentifyer3.properties

```
1
2  consumerKey=Q4s2bIwpEEFcYT4zOtjgWlNOM
3  consumerSecret=2YN2Mjl662s6HYSJhaYrkvBpVMppM5rHiWWeAsSQZabT0bCYbM
4  accessToken=1238876335789015042-fxtakDFzXhFl0XC8cAK6qrKdMNQSDn
5  accessTokenSecret=UiUMPukyOccDWd5oz21ccinZTgQuQVWWk4O9HTAnIlxzI
```

Figura 7.12. Contenido del archivo de propiedades de claves 1

genderIdentifyer1.properties | genderIdentifyer2.properties | genderIdentifyer3.properties

```
1
2  consumerKey=ebYGA2aOftitQKQ52c8GxcDHG
3  consumerSecret=LcsknfGVbUNLUElQpJa4fTZWmvAzaNbIgxcC8MDz394Hr1PRdH
4  accessToken=1238876335789015042-NYeD8pyCWb7kGHUHM4oToHzJKgR1N6
5  accessTokenSecret=VLXDXkiFY5R8A26WjSrJTnTKhdiIH3xRvAmHu3WqIaG6f
```

Figura 7.13. Contenido del archivo de propiedades de claves 2

genderIdentifyer1.properties | genderIdentifyer2.properties | genderIdentifyer3.properties

```
1
2  consumerKey=XnnbCggOghiNDJEAsQsxgGacn
3  consumerSecret=MhwIZ04bGGFD6C0dkZ0pBfutHR7zKPinQxvRQu73VhPogFnJQu
4  accessToken=1238876335789015042-uxRLAz8xlpnG1e8g1OkCPZNHOTMrNV
5  accessTokenSecret=sxoeQr9xqofVIgiTxE6JZDAtYqzCZenUAc6KIAh8kSvDG
```

Figura 7.14. Contenido del archivo de propiedades de claves 3

De la gestión posterior de estos archivos se encarga la clase *ApplicationProps.java*

Además, existe una clase *Printer.java* que es la encargada de crear los archivos de salida de la aplicación (archivos de usuarios, archivos de tweets, archivos de usuarios descartados y archivo de características (dataset)):

- *[año][mes[día]-[horas][minutos][segundos] BoyUsersList*
- *[año][mes[día]-[horas][minutos][segundos] GirlUsersList*
- *[año][mes[día]-[horas][minutos][segundos] BoyTweetsList*
- *[año][mes[día]-[horas][minutos][segundos] GirlTweetsList*
- *[año][mes[día]-[horas][minutos][segundos] DiscardBoyUsersList*
- *[año][mes[día]-[horas][minutos][segundos] DiscardGirlUsersList*
- *[año][mes[día]-[horas][minutos][segundos] linguisticFeatures.txt*

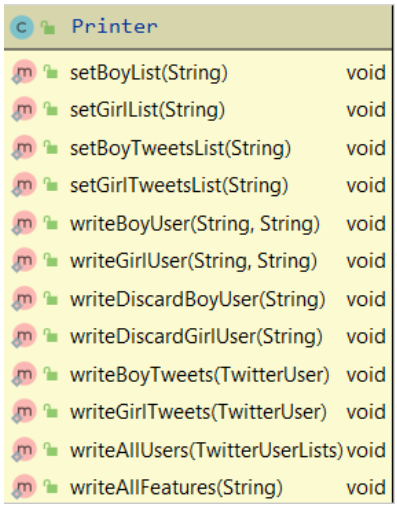


Figura 7.15. Métodos de clase *Printer*

La estructura principal del proyecto excluyendo el controlador es la siguiente:

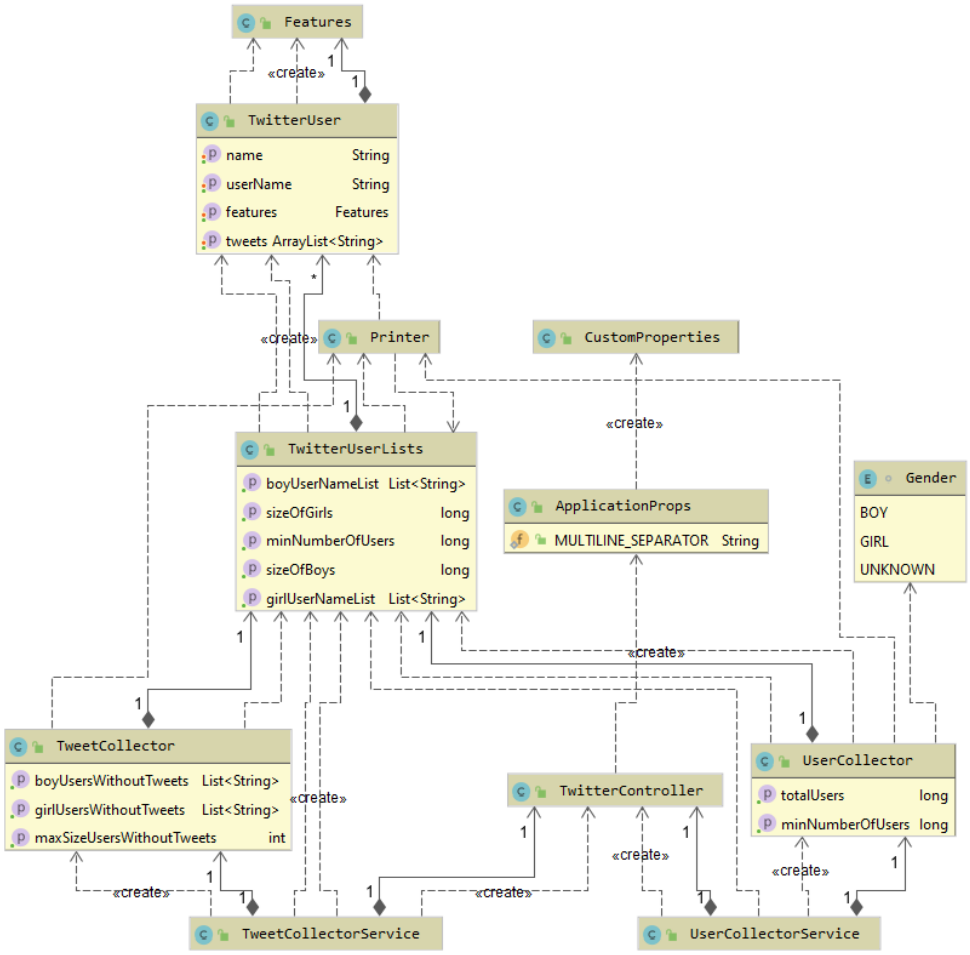


Figura 7.16. Esquema general de las relaciones de clases del proyecto

El detalle del funcionamiento de estos componentes del componente principal se explicará en detalle a continuación.

7.2.1.1 Elementos base

Como componente básico de la aplicación se encuentra la clase *TwitterUser.java*.

Esta clase representa a un usuario de Twitter en el sistema y tiene como atributos el nombre del usuario, su nickname, una lista de tweets y un objeto *Features*. Cuenta con los métodos necesarios para gestionar su propia información.

Para gestionar de forma separada las listas de usuarios hombres y mujeres se usa la clase *TwitterUserLists.java*. Esta clase tiene como atributos una lista de usuarios identificados con el género de masculino y otra lista con usuarios identificados con el género femenino. Tiene todos los métodos necesarios para gestionar la información de estas listas además del método con el que se generan todas las características de cada usuario con las que se formará finalmente el dataset.

7.2.1.2 Conexión con Twitter - Librería twitter4j

En la aplicación se emplea la librería de Java *twitter4j* para conectar con la API de Twitter y usar sus funcionalidades. Para usarla se crea la clase *TwitterController.java* que cuenta con un atributo de la clase *Twitter* a través del cual se realizan todas las operaciones deseadas de la API. Este atributo se debe configurar con un objeto *ConfigurationBuilder,* en él se establece el modo extendido (para devolver los tuits completos, 280 caracteres en lugar de los antiguos 140) y los tokens obtenidos para la aplicación, los cuales están guardados en los ficheros de propiedades. Después con la clase *TwitterFactory* se consigue la instancia de la clase *Twitter* que se usará.

Hay que destacar dos métodos principales:

- *List<Status> searchUsers (String searchKey)*

  Encargado de hacer una búsqueda de una palabra (*searchKey*) en los tweets más recientes del timeline de Twitter. Establece el idioma en español, la ubicación en Madrid y un radio de búsqueda de 500km. Devuelve los 100 últimos tuits.

- *List<Status> searchTweets(String userName, int page)*

  Devuelve todos los tuits de la página *page* de un usuario (*userName*) de Twitter. Las páginas se dividen de 200 en 200 tuits.

7.2.1.3 Extractor de usuarios

Las clases que forman este componente son *UserCollectorService.java*, la cual únicamente interactúa con la API de Twitter para hacer las consultas necesarias y *UserCollector.java*, que se encarga de procesar la información devuelta por la API para extraer los usuarios. Los métodos principales son *startCollection* y *collectUsers* respectivamente.

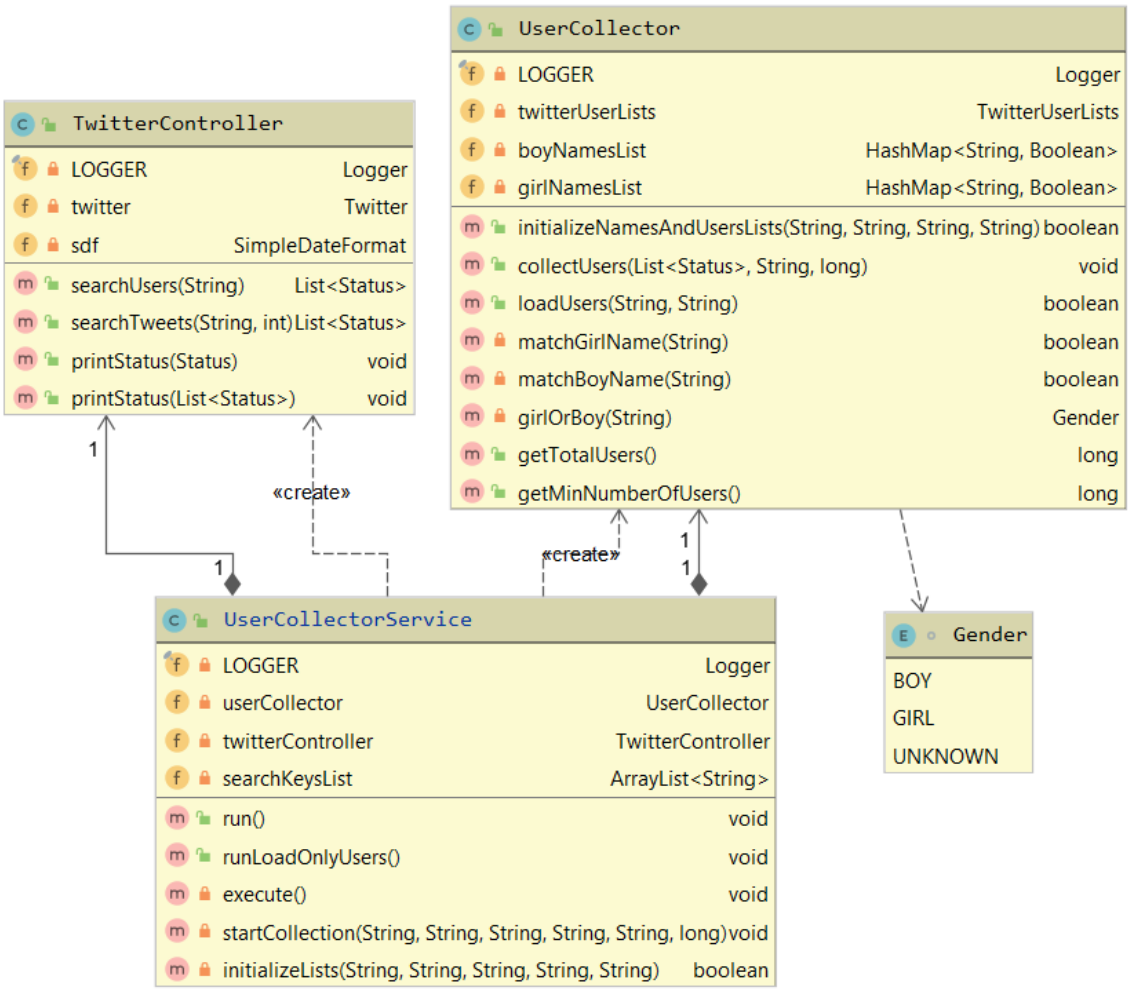


Figura 7.17. Relación de las clases *UserCollector*, *UserCollectorService* y *TwitterController*

En estos métodos, de cada tuit devuelto en la consulta se extrae el usuario y el nombre del mismo, además si se trata de un retuit también se obtiene el usuario y el nombre de la persona que publico el tuit original.

Si el usuario extraído todavía no se había procesado en la aplicación, su nombre se compara con la lista de nombres de hombres y la lista de nombres de mujeres. Si es posible determinar el género del usuario se pasa a verificar si el usuario tiene al menos 500 tuits publicados (suficientes para extraer más adelante) y menos de 5000 seguidores (para evitar cuentas con mucha repercusión).

Si se cumplen todos los requisitos se guarda el usuario y su nombre en el fichero correspondiente.

La extracción de usuarios finaliza cuando se alcanza el número deseado de usuarios hombres y mujeres o bien cuando se produzca una excepción por parte de la conexión con Twitter del tipo *Rate limit exceeded.*

En este caso no se rotan las claves de conexión con las API de Twitter ya que volver a ejecutar las mismas búsquedas inmediatamente en el timeline devolvería prácticamente los mismos los usuarios. Por ello es mejor que una vez se pare la recolección de usuarios se inicie la recolección de sus tuits y cuando esta finalice entonces volver a ejecutar la extracción de los usuarios.

7.2.1.4 Extractor de mensajes (tuits)

Este componente está formado por las clases *TweetCollectorService.java* y *TweetCollector.java.* La clase de servicio únicamente se encarga de las interacciones con la API de Twitter mientras que es la clase *TweetCollector* quien se encarga de gestionar la información de los tuits recibidos. Los métodos principales son *collectTweets* en clase de servicio y *collectBoyTweets* y *collectGirlTweets* en la otra clase.

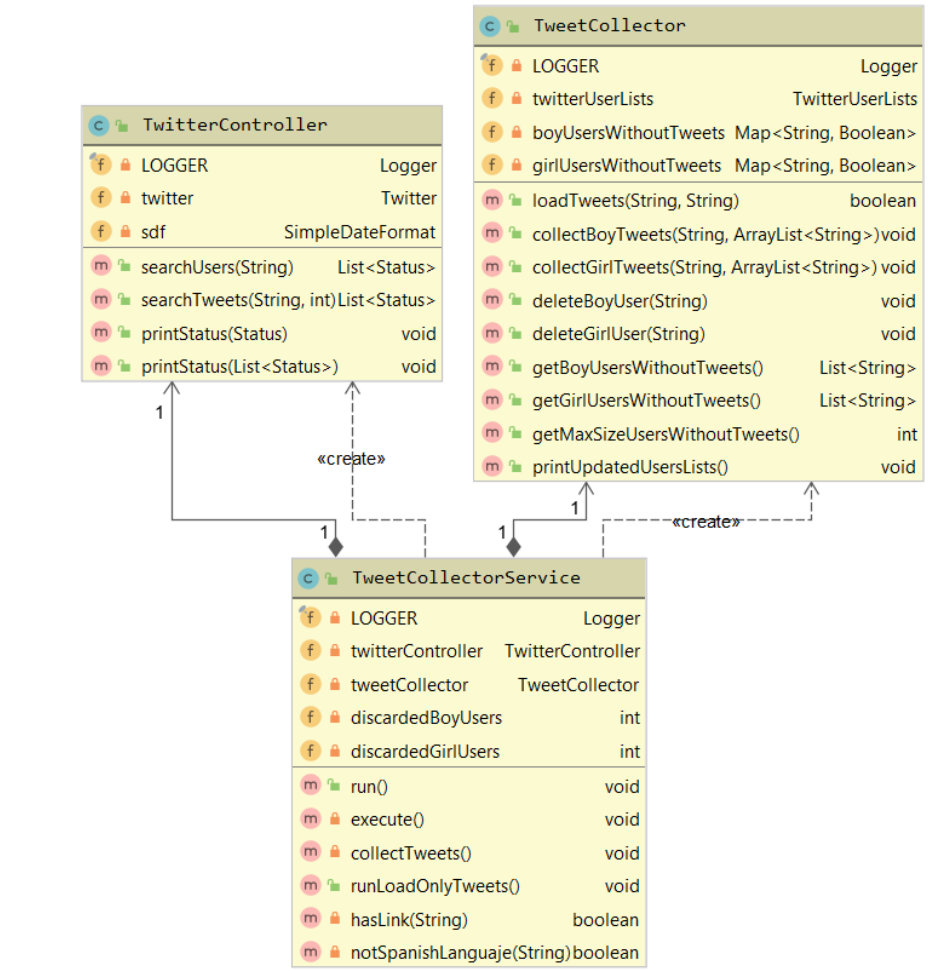


Figura 7.18. Relación de las clases *TweetCollector*, *TweetCollectorService* y *TwitterController*

Con estos métodos se extraen 200 tuits de un usuario que ya esté cargado en el sistema por cada consulta. En muchas ocasiones es necesario realizar varias consultas por usuario ya que suelen tener retuits, los cuales no se tienen en cuenta como un tuit propio. Así se rescatan los 100 últimos tuits de un usuario, pero se tiene que cumplir con tres criterios:

- Si no se consiguen recolectar los 100 tuits necesarios en 10 consultas el usuario se descarta.
- Si en más de un 65% de los tuits de la lista existe algún enlace, el usuario se descarta (evitar usuarios que hagan publicidad, cuyo lenguaje podría estar condicionado).

- Si se encuentra algún carácter o palabra de otro idioma que no sea el español en algún tuit (apéndice I), el usuario se descarta.

Si finalmente se consigue una lista de tuits válida, esta se le asigna a su usuario y los tuits se guardan en el fichero que corresponda.

Tras finalizar la ejecución (una vez que se hayan extraído los tuits de todos los usuarios en el sistema) se actualizan las listas de usuarios y los archivos correspondientes eliminando los usuarios descartados.

7.2.1.5 Procesador del texto

Formado por las clases *Features.java* y *FunctionWords.java*

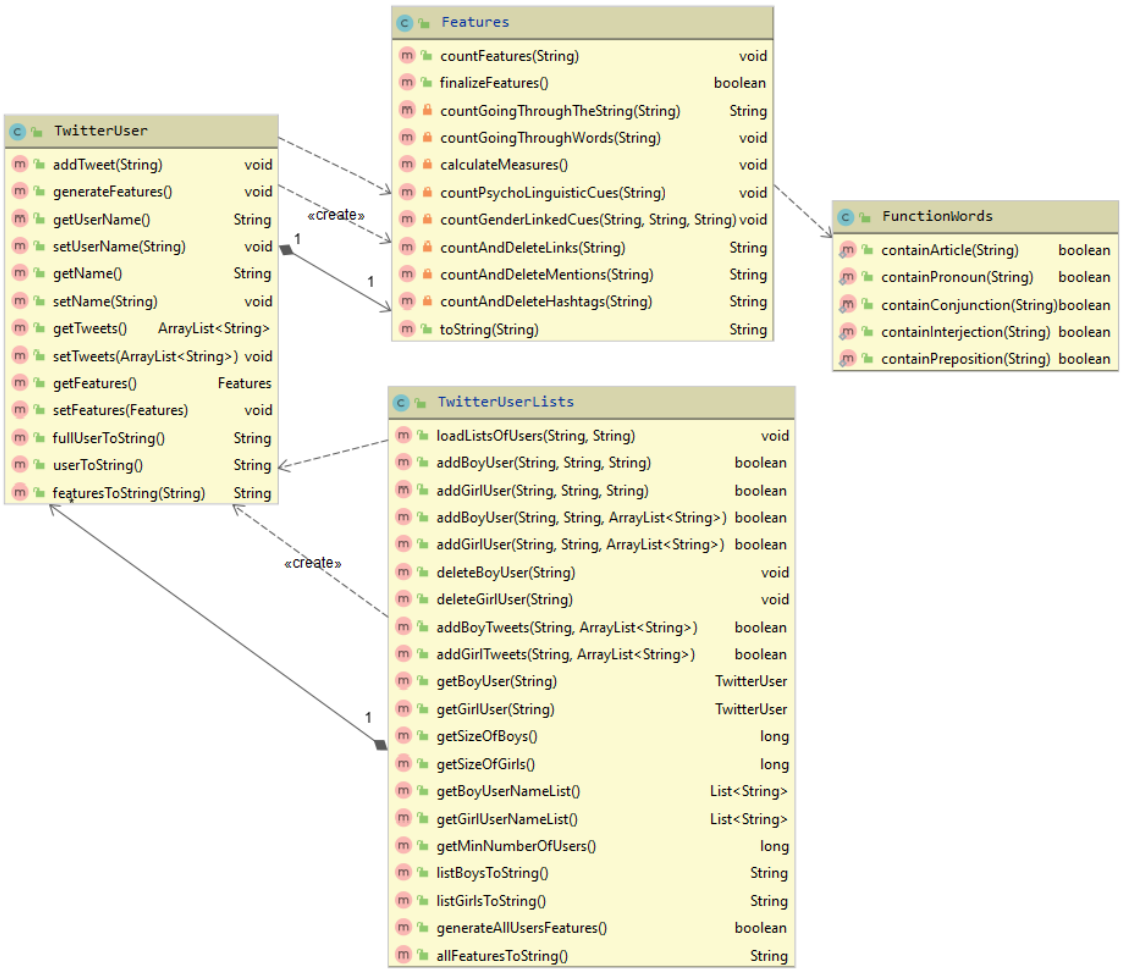


Figura 7.19. Relación de las clases *Features*, *FunctionWords*, *TwitterUser* y *TwitterUserList*

FunctionWords es una clase estática con varios HashMap que se usan para identificar palabras. Hay mapas de pronombres, conjunciones, interjecciones y preposiciones.

La clase *Features* cuenta con un atributo por cada característica del texto que se debe cuantificar (apéndices A-H).

Los métodos principales de esta clase son *countFeatures(String tweet)* que se encarga de cuantificar todas las características de un tweet de un usuario del sistema; y *finalizeFeatures()* cuya funcionalidad es calcular definitivamente cada característica y calcular las características que son métricas y necesitan que el resto hayan sido calculadas previamente (apéndice D).

7.2.2 Script R

Este componente es el encargado de crear el clasificador basado en una Maquina de Vectores de Soporte. Para ello se usa el lenguaje R puesto que ofrece librerías que ya contienen algunos algoritmos de clasificación, en concreto el que necesitamos. Todo el script está comentado para una mejor comprensión de los pasos a seguir, se puede consultar en los materiales adjuntos a este trabajo (*GenderIdentifyer/RProject/GenderIdentifyer.R*)

La primera parte del script consiste en cargar los datos del dataset previamente generado. Una vez cargados se normalizan para que todas las columnas (características) se encuentre en el rango de valores [0,1], ya que el algoritmo SVM es sensible a variaciones grandes entre los rangos de las diferentes variables.

```
26 #Cargamos los datos del dataset, stringAsFactor para convertir la variable de repuesta (M,F)
27 data <- read.table(file="linguisticFeatures.txt", header = TRUE, sep = " ", dec = ".",
28                    stringsAsFactors =  TRUE);
29
30 str(data); #Formato de los datos
31 head(data); #Primeros datos
32
33
34
35
36 #Lista de datos numéricos
37 listIsnum <- sapply(data, is.numeric)
38
39 #Normalizamos todos los datos del dataset (los numéricos)
40 normalizedData <-data
41 normalizedData[,listIsnum] <- lapply(data[,listIsnum], normalize)
```

Figura 7.20. Parte del script en R que carga y normaliza los datos

Posteriormente se eliminan las columnas que contengan algún valor sin cuantificar (*NA* – valor vacío) y también aquellas que tengan una varianza próxima a cero (para evitar que afecten a la predicción del sistema, pues no aportan valor).

```
#Se eliminan todas las columnas que tengan algun NA en sus registros
colsWithNaNames <- colnames(normalizedData)[!!colSums(is.na(normalizedData))]
colsWithNaNames
colsWithNa <- colSums(is.na(normalizedData)) == 0
colsWithNa
normalizedData <- normalizedData[ , colSums(is.na(normalizedData)) == 0]


#Se eliminan todas las columnas que tengan una varianza próxima a 0
colsWithZeroVarNames<- colnames(normalizedData[,nearZeroVar(normalizedData)])
colsWithZeroVarNames

colsWithZeroVar <- nearZeroVar(normalizedData)
normalizedData <- normalizedData[,-nearZeroVar(normalizedData)]
```

Figura 7.21. Parte del script en R que elimina columnas sin valores y con varianza próxima a 0

Con los datos ya procesados se crean dos particiones de los mismos, una que representa el 80% de los datos y se usará para el entrenamiento; y otra del 20% restante para testearlo posteriormente.

```
#Se crea una partición con un 80% de los datos para el entrenamiento y un 20% para la posterior validación
set.seed(123)
partition <- createDataPartition(y = normalizedData$gender, p = 0.8, list = FALSE, times = 1)

# Datos entrenamiento
trainData <- normalizedData[partition, ]

# Datos validación
testData <- normalizedData[-partition, ]
```

Figura 7.22. Parte del script en R que crea las particiones con los datos

Tras esto se buscan los mejores hiperparámetros para el sistema, que tendrá un kernel radial (en este caso hiperparámetros *coste* y *gamma)*. Esto se consigue mediante validación cruzada gracias a la función *tune*, que va probando diferentes valores para los hiperparámetros obteniendo el error de validación para cada una de las combinaciones. Es la parte que más tarda del proceso pudiendo consumir más de una hora en completarse.

```
104 #Ajuste del modelo radial (hiperparámetros óptimos mediante validación cruzada)
105 set.seed(325)
106 tuning <- tune(svm, gender ~ ., data = trainData,
107                kernel = "radial",
108                ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15),
109                              gamma = c(0.01, 0.1, 1, 5, 10)),
110                scale = TRUE)
111
112 summary(tuning)
```

Figura 7.23. Parte del script en R que realiza la validación cruzada

Finalmente, se obtiene el mejor modelo (aquel que cuenta con los hiperparámetros óptimos) y se prueba con los datos de test, comprobando así su precisión. Este modelo se guarda en un archivo *.RData*.

```
129 #Se extrae del mejor modelo
130 svmRadialModel <- tuning$best.model
131
132 #Tambien se puede generar el modelo con los mejores parámetros (bestModel)
133 #svmRadialModel <- svm(gender ~ ., data = trainData,
134 #                      kernel = "radial",
135 #                      cost = 1,
136 #                      gamma = 0.01,
137 #                      scale = TRUE)
138
139 summary(svmRadialModel)
140
141
142
143
144 #Se evalua el modelo
145 #Se generan las predicciones
146 prediction <- predict(svmRadialModel,testData)
```

Figura 7.24. Parte del script en R que con genera el mejor modelo y realiza las predicciones

Capítulo 8. Resultados

El primer logro del trabajo es la generación de un dataset con las características analizadas y cuantificadas de 10000 usuarios, 5000 mujeres y 5000 hombres extrayendo sus 100 últimos tweets.

20210202-105502 linguisticFeatures.txt

```
gender numberOfCharacters numberOfLetters numberOfUpper numberOfDigital numberO
M 4047 0.80826 0.00149 0.00322 0.17866 0 0.00025 0 0.15 0.22 0 805 408 4.07951
M 6292 0.76908 0.03449 0.00414 0.16688 0 0.00032 0.00207 0.25 1.27 0.02 1116 51
M 7010 0.78189 0.07019 0.00371 0.16848 0 0.00329 0.00428 0.52 0.03 0.5 1216 535
M 3798 0.78621 0.03845 0.00395 0.16378 0 0.00527 0.00501 0.49 0.41 0 685 387 4.
M 7659 0.77152 0.0346 0.00444 0.15956 0 0.02403 0.00196 0.21 0.59 0.59 1340 599
M 12049 0.77501 0.03868 0.00631 0.17255 0 0.00025 0.00241 0.32 1.88 0.02 2165 8
M 4012 0.77967 0.02194 0.00374 0.16975 0 0.00424 0.00125 0.22 0.09 0.03 739 369
M 13494 0.80236 0.0215 0.0006 0.17527 0 0.00023 0 0 1.44 0 2455 855 4.41345 0.3
M 4224 0.78788 0.02889 0.009 0.16975 0 0 0.00356 0.06 1.07 0 799 394 4.21277 0.
M 3806 0.68209 0.03994 0.00316 0.14714 0 0.00342 0.00106 0.31 0.99 0.02 622 309
M 5694 0.7731 0.04953 0.00317 0.16737 0 0.00053 0.00141 0.43 0.74 0 1015 484 4.
M 6935 0.76525 0.02437 0.00073 0.17304 0 0 0.00015 0.01 1.25 0 1256 517 4.2293
M 11106 0.78283 0.03098 0.00253 0.16532 0 0.00298 0.00064 0.61 1.6 0.07 1903 78
M 5624 0.80477 0.03557 0.00907 0.15648 0 0.00036 0.00125 0.08 1.6 0 978 506 4.6
M 7568 0.80088 0.03291 0.00437 0.16068 0 0.00119 0.0004 0.24 1.62 0 1297 645 4.
M 16388 0.79077 0.02228 0.00214 0.18191 0 0.00068 0.00037 0.15 2.12 0 3129 957
M 5214 0.7547 0.0259 0.02379 0.1636 0 0.00077 0.01075 0.43 0.49 0.01 934 470 4.
```

Figura 8.1. Archivo que contiene el conjunto de datos final

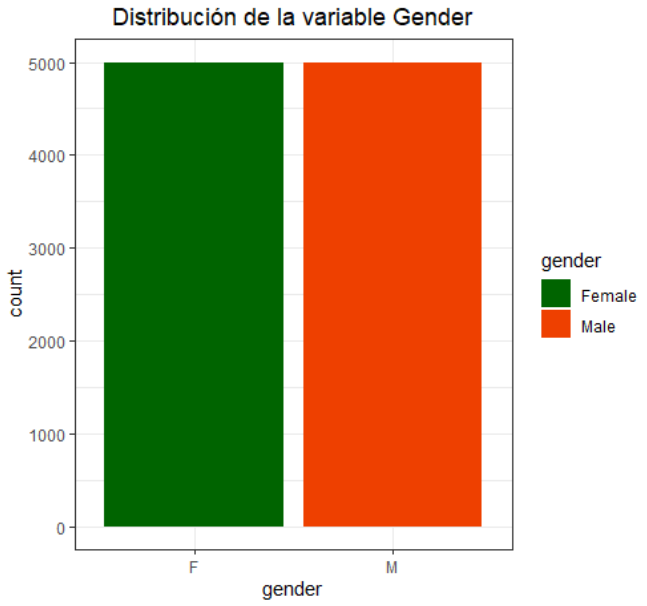


Figura 8.2. Gráfica de distribución de la variable respuesta “Gender”

Este dataset esta creado con las variables que se pueden consultar más en detalle en los apéndices (apéndices A-H), se trata un total de 63 variables (1 de respuesta y 1 que se eliminará del dataset) con los siguientes tipos:

```
> str(data); #Formato de los datos
'data.frame':   10000 obs. of  63 variables:
 $ gender                   : Factor w/ 2 levels "F","M": 2 2 2 2
 2 ...
 $ numberOfCharacters       : int  4047 6292 7010 3798 7659 12049
 4224 3806 ...
 $ numberOfLetters          : num  0.808 0.769 0.782 0.786 0.772 .
 $ numberOfUpper            : num  0.00149 0.03449 0.07019 0.03845
 ...
 $ numberOfDigital          : num  0.00322 0.00414 0.00371 0.00395
 0.00631 0.00374 0.0006 0.009 0.00316 ...
 $ numberOfWhiteSpace       : num  0.179 0.167 0.168 0.164 0.16 ..
 $ numberOfTab              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ numberOfLineBreaker      : num  0.00025 0.00032 0.00329 0.00527
 ...
 $ numberOfSpecialCharacters: num  0 0.00207 0.00428 0.00501 0.001
 0.00125 0 0.00356 0.00106 ...
 $ numberOfLinks            : num  0.15 0.25 0.52 0.49 0.21 0.32 0
 0.31 ...
 $ numberOfMentions         : num  0.22 1.27 0.03 0.41 0.59 1.88 0
 1.07 0.99 ...
 $ numberOfHashtags         : num  0 0.02 0.5 0 0.59 0.02 0.03 0 0
 $ numberOfWords            : int  805 1116 1216 685 1340 2165 739
 622 ...
 $ numberOfDifferentWords   : int  408 513 535 387 599 890 369 855
```

Figura 8.3. Formato de algunas de las variables del dataset

Tras ejecutar el script de R sobre este dataset, se han podido extraer los siguientes resultados:

- Tras una primera ejecución, se observó que la variable que representa el número de caracteres tabulador '\t' que si se cuantificó según en el artículo *Author gender identification from text* [3], tomaba siempre un valor NA, es decir, no se cuantificaba. Esto llevo a tomar posteriormente la decisión de eliminar la misma directamente de la aplicación encargada de generar el dataset, ya que si no se encontró ni una sola aparición en esta muestra, que puede considerarse bastante significativa, no tiene sentido que se mantenga para futuras generaciones de datos.

```
> colsWithNaNames <- colnames(normalizedData)[!!cols
a))]
> colsWithNaNames
[1] "numberOfTab"
>
```

Figura 8.4. Columnas con valores sin cuantificar

- En este caso, existen algunas variables que tienen una varianza igual o cercana a 0 (casi siempre los mismos valores en cada observación). A diferencia del caso anterior no se decidió, tras comprobarlo, eliminarlas del generador del dataset ya que en futuras muestras esto podría cambiar, pero si se excluyen en el entrenamiento de este modelo ya que estas no aportan ningún valor a la hora de generarlo (gestionado por el propio script R).

```
> colsWithZeroVarNames
[1] "singleQuotes"              "doubleQuotes"
[3] "colons"                    "openQuestions"
[5] "multipleQuestionsMarks"    "openExclamations"
[7] "multipleExclamationsMarks" "interjections"
```

Figura 8.5. Variables con varianza igual o cercana a 0

- Tras el entrenamiento del modelo, se obtuvo que los hiperparámetros óptimos (los que más reducen el error de validación) para el ajuste del mismo eran *coste = 1* y *gamma = 0.01.* Este proceso consumió más de tres horas, habiéndose comprobado mediante validación cruzada los valores *coste = 0.001, 0.01, 0.1, 1, 5, 10, 15* y *gamma = 0.01, 0.1, 1, 5, 10*.

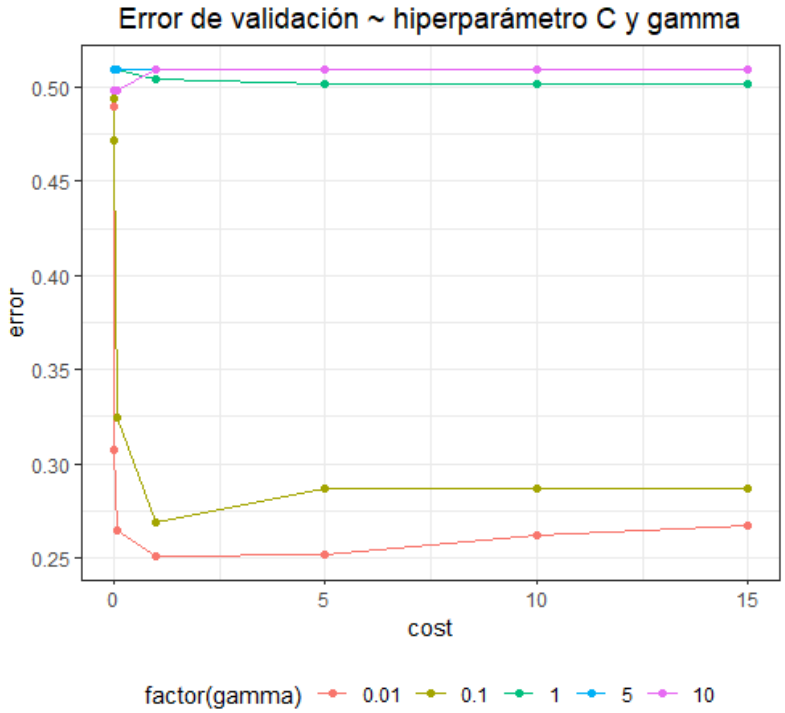


Figura 8.6. Gráfica del error de validación en función de los hiperparámetros

```
> summary(tuning)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1  0.01

- best performance: 0.25125
```

Figura 8.7. Resumen de la función tune

- Con estos valores de los hiperparámetros y tras testear el modelo, se obtuvo una precisión en el mismo del 75,7%.

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  5054

 ( 2532 2522 )


Number of Classes:  2

Levels:
 F M
```

Figura 8.8. Resumen del mejor modelo

Arrojando un 24,3% de error en la clasificación, habiéndose fallado 245 casos para el género masculino y 241 casos para el género femenino.

```
Confusion Matrix and Statistics

          Reference
Prediction   F   M
         F 759 245
         M 241 755

               Accuracy : 0.757
                 95% CI : (0.7376, 0.7757)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.514

 Mcnemar's Test P-Value : 0.8918

            Sensitivity : 0.7590
            Specificity : 0.7550
         Pos Pred Value : 0.7560
         Neg Pred Value : 0.7580
             Prevalence : 0.5000
         Detection Rate : 0.3795
   Detection Prevalence : 0.5020
      Balanced Accuracy : 0.7570

       'Positive' Class : F
```

Figura 8.9. Matriz de confusión del mejor modelo

Capítulo 9. Conclusiones y trabajo futuro

Gracias a los resultados arrojados por el modelo SVM, el cual se entrenó con el dataset generado por la aplicación, se puede decir que se trata de un buen conjunto de datos para el entrenamiento de este tipo de modelos.

Si nos fijamos en la precisión del estudio en el que está basado este trabajo (*Author gender identification from text* [3]), esta asciende hasta el 82,23% para textos en inglés extraídos de correos electrónicos privados y habiendo tenido en cuenta un total de 93 características diferentes.

Se podría por tanto afirmar que haber obtenido una precisión del **75,7%** analizando un total de 61 variables diferentes, en un idioma distinto y para otro contexto de los mensajes, es un buen dato, aunque sin duda mejorable.

También cabe destacar el balanceo del dataset, aunque la clase positiva es el género femenino habiendo fallado en su predicción 4 veces menos, se puede afirmar que se trata de un conjunto de datos equilibrado y que no tiene un sesgo hacia ninguna de las variables de respuesta.

Como trabajo futuro, se podrían ampliar estas características a analizar, por ejemplo, añadiendo las características analizadas gracias a LIWC [19], que podrían aumentar significativamente la precisión de los sistemas. Analizar el tipo de emoticonos usados por cada género; o el número de retuits y favoritos que reciben también podrían ser características interesantes de analizar.

Apéndices

Características utilizadas

Apéndice A

Características basadas en caracteres	
1	Nº de caracteres (C)
2	Nº de letras / C
3	Nº de mayúsculas / C
4	Nº de dígitos / C
5	Nº de espacios en blanco / C
6	Nº de saltos de línea / C
7	Nº de caracteres especiales / C
8	Media de caracteres por tweet

Apéndice B

Características propias de Twitter	
9	Nº de enlaces / Nº de tweets
10	Nº de menciones / Nº de tweets
11	Nº de hashtags / Nº de tweets

Apéndice C

Características basadas en palabras	
12	Nº de palabras (P)
13	Nº de palabras diferentes
14	Tamaño medio de palabra
15	Riqueza de vocabulario / P
16	Palabras mayores de 6 caracteres / P
17	Palabras pequeñas / P
18	Media de palabras por tweet / P

Apéndice A1

Caracteres especiales	
1	|
2	·
3	$
4	%
5	&
6	¬
7	(
8	)
9	=
10	^
11	[
12	*
13	+
14	]
15	{
16	}
17	-
18	_
19	>
20	<
21	ª
22	\
23	º
24	€

Riqueza de vocabulario: Nº de palabras diferentes en el texto.

Apéndice D

Características métricas sobre la riqueza de vocabulario	
19	Hapax legomena / P
20	Hapax dislegomena / P
21	Yules' K
22	Simpson's D
23	Sichel's S
24	Honore's R
25	Entropía

Hapax Dislegomena: palabras que aparecen solo dos veces.
Hapax Legomena: palabras que aparecen solo una vez.

Apéndice D1

V: Nº palabras diferentes.
Vi: Nº palabras diferentes que aparecen i veces.
N: Nº total de palabras

$$\text{Yules K} = 10^4\left(-\frac{1}{N}+\sum_{i=1}^{V} V_i\left(\frac{i}{N}\right)^2\right)$$

$$\text{Simpsons D} = \sum_{i=1}^{V} V_i \frac{i}{N}\frac{i-1}{N-1}$$

$$\text{Sichels S} = \frac{\text{count of Hapax Dislegomena}}{V}$$

$$\text{Honores R} = \frac{100 log_{10} N}{1-\frac{\text{count of Hapax Legomena}}{V}}$$

$$\text{Entropy} = \sum_{i=1}^{N} V_i\left(-log_{10}\frac{i}{N}\right)\frac{i}{N}$$

Apéndice E

Características sintácticas	
26	Nº de comillas simples / C
27	Nº de comillas dobles / C
28	Nº de comas / C
29	Nº de puntos / C
30	Nº de puntos suspensivos/ C
31	Nº de dos puntos / C
32	Nº de puntos y comas / C
33	Nº de interrogación apertura / C
34	Nº de interrogación cierre / C
35	Nº de exclamación apertura / C
36	Nº de exclamación cierre / C
37	Nº de multiples interrogaciones / C
38	Nº de múltiples exclamaciones / C

Apéndice F

Características palabras funcionales	
39	Nº de artículos / P
40	Nº de pronombres / P
41	Nº de conjunciones / P
42	Nº de interjecciones / P
43	Nº de preposiciones / P

Apéndice G

Características palabras psicolingüísticas	
44	Nº de negaciones
45	Nº de emociones positivas
46	Nº de emociones negativas
47	Nº de ansiedad
48	Nº de ira
49	Nº de tristeza
50	Nº de percepciones
51	Nº de tentativas
52	Nº de certezas
53	Nº de inhibiciones
54	Nº de afirmaciones

Apéndice G1

Negaciones	NO, NUNCA
Emociones positivas	AMOR, BONITO, BONITA, AGRADABLE
Emociones negativas	HERIDO, FEO, FEA, DESAGRADABLE
Ansiedad	PREOCUPADO, TEMEROSO, NERVIOSO, PREOCUPADA, TEMEROSA, NERVIOSA
Ira	ODIA, ODIO, MATA, MATO, MOLESTO, MOLESTA
Tristeza	LLORA, LLORO, DOLOR, TRISTE
Percepciones	PENSA, PIENSA, PIENSO, SABE, SÉ, CONSIDER
Tentativas	QUIZÁS, ADIVINA, ADIVINO, ACASO
Certezas	SIEMPRE, NUNCA
Inhibiciones	BLOQUEA, BLOQUEO, RESTRINGI, RESTRINJ, DETENE
Afirmaciones	OK, SI, OKEY, OKAY

Apéndice H

Características palabras vinculadas al género	
55	Nº de adjetivos afectivos
56	Nº de insultos
57	Nº de exclamaciones
58	Nº de evasivas
59	Nº de adverbios de intensidad
60	Nº de adjetivos críticos
61	Nº de verbos de incertidumbre

Apéndice H1

Adjetivos afectivos	ADORABLE, ENCANTADOR, ENCANTADORA, DULCE, BONITO, BONITA, DIVINO, DIVINA
Insultos	GILIPOLLAS, CABRON, CABRÓN, CABRONA, PUTA, PUTO, SUBNORMAL, IDIOTA, IMBÉCIL, IMBECIL, TONTO, TONTA
Exclamaciones	JODER, HOSTIA, OSTIA, COÑO
Evasivas	POSIBLEMENTE, TAL VEZ, UNA ESPECIE DE
Adverbios intensidad	REALMENTE, MUY, BASTANTE, ESPECIALMENTE, BASTANTE, DEMASIADO
Adjetivos críticos	DISTRAIDO, DISTRAIDA, MOLESTO, MOLESTA, AMABLE, GENIAL
Verbos incertidumbre	PREGUNTO, CONSIDERO, SUPONGO, CREO

Palabras o caracteres en otros idiomas

Apéndice I

Palabras en gallego	HOXE, NOITE, ESQUERDA, GRAZAS, DEREITA, XEITO, COITADO, BOS DÍAS, BOAS NOITES, BOS DIAS, NOITES, FACEM, ISSO
Caracteres en catalán/francés	Â, Ç, À, Ê, È, Ô, Ò
Palabras en euskera	ASKO, KAIXO, GABON, EGUN ON, ZABALIK TXARRA, ITXITA
Palabras en inglés	REALLY, BECAUSE, FIRST, THAN, WITH, THAT, OTHER, WANT, FROM, THEN

Bibliografía

[1] Estilometría, Wikipedia. Disponible en:

https://es.wikipedia.org/wiki/Estilometría

[2] Andrés González, “La importancia de limpiar, seleccionar y transformar los datos”. Disponible en:

https://cleverdata.io/limpiar-seleccionar-transformar-datos/

[3] Cheng, Na & Chandramouli, Rajarathnam & Subbalakshmi, K. (2011). “Author gender identification from text”. Digital Investigation. 8. 78-88. Disponible en:

https://www.researchgate.net/publication/220346182_Author_gender_identification_from_text

[4] Machine Learning, Aprendizaje automático, Wikipedia. Disponible en:

https://es.wikipedia.org/wiki/Aprendizaje_automático

[5] Saeid Safavi, Martin Russell, Peter Jančovič, enero 2018. “Automatic Speaker, Age-group and Gender Identification from Children’s Speech”. Disponible en:

https://www.researchgate.net/publication/322336038_Automatic_Speaker_Age-group_and_Gender_Identification_from_Children's_Speech

[6] A.S. Tolba, julio 2001. “Invariant Gender Identification”. Disponible en:

https://www.researchgate.net/publication/220137017_Invariant_Gender_Identification

[7] Shereen Husseina, Mona Farouka, ElSayed Hemayedab, julio 2019. “Gender identification of egyptian dialect in twitter”. Disponible en:

https://www.sciencedirect.com/science/article/pii/S1110866518302044

[8] François Rebaudo, mayo 2020. “Aprender R: iniciación y perfeccionamiento”. Disponible en:

https://myrbooksp.netlify.app/

[9] Joaquín Amat Rodrigo, abril 2017. “Máquinas de Vector Soporte (Support Vector Machines, SVMs)”. Disponible en:

https://rpubs.com/Cristina_Gil/SVM

[10] Cristina Gil Martínez, junio 2018. “MÁQUINAS DE VECTOR SOPORTE”. Disponible en:

https://rpubs.com/Joaquin_AR/267926

[11] Java, Lenguaje Java. Disponible en:

https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)

[12] JavaFX, Getting Started with JavaFX. Disponible en:

https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784

[13] Twitter API, Información sobre las API de Twitter. Disponible en:

https://developer.twitter.com/en/docs

https://help.twitter.com/es/rules-and-policies/twitter-api

[14] Twitter4j, twitter4j. Disponible en:

http://twitter4j.org/en/

[15] R, The R Project for Statistical Computing. Disponible en:

https://www.r-project.org/

https://www.unir.net/ingenieria/revista/lenguaje-r-big-data/

[16] Intellij IDEA, JetBrains. Disponible en:

https://www.jetbrains.com/es-es/idea/

https://es.wikipedia.org/wiki/IntelliJ_IDEA

[17] R Studio, R Studio web. Disponible en:

https://rstudio.com/

[18] Pro-sentence. Definitions.net. Disponible en:

https://www.definitions.net/definition/PRO-SENTENCE

[19] Nairán Ramírez-Esparza, James W. Pennebaker, Florencia Andrea García, Raquel Suriá. “La Psicología del Uso de las Palabras: Un Programa de Computadora que Analiza Textos en Español”. Disponible en:

https://labclab.psychology.uconn.edu/wp-content/uploads/sites/1167/2015/04/health1.pdf

Anexos

Manual de uso

Twitter API

En un primer lugar es necesario registrarse en la plataforma de Twitter, para ello se debe disponer de un correo electrónico y un número de teléfono válidos, ya que serán necesarios para verificar más adelante la cuenta.

Con el registro completado y la cuenta verificada, el siguiente paso sería acceder a la página de desarrolladores de Twitter https://apps.twitter.com. En esta página se debe crear una nueva aplicación, sin embargo, si la cuenta no es de desarrollador, Twitter solicita cierta información para poder crear la cuenta y la aplicación.

Primero Twitter pregunta cual es razón para usar las herramientas de desarrollador, la opción que se eligió fue la de estudiante.

Después de esto se hacen una serie de preguntas muy específicas sobre la aplicación, el uso que se le dará a la API y el tipo y tratamiento que se les dará a los datos extraídos. Las respuestas se dan en inglés y tienen una longitud mínima, no pueden ser aleatorias ni sin sentido ya que se revisarán para dar o no el acceso a la cuenta de desarrollador.

Tras enviar las respuestas, Twitter después de revisarlas envía un correo de confirmación. Después de esto la cuenta de desarrollador ya estaría activa.

Para tener acceso a las claves necesarias para el uso de la *API Search* y la *API REST* se crea la aplicación *"GenderIdentifier"*. Es necesario proporcinar el nombre, una descripción y un sitio web para la aplicación; y posteriormente aceptar los términos y condiciones. Una vez hecho se generan unos tokens (*Consumer Key*, *Consumer Secret*, *Access Token*, *Access Token Secret*), los cuales no caducan hasta que el usuario los revoque y pueden actualizarse también en cualquier momento.

Posteriormente se crean otras dos aplicaciones similares para obtener otros dos conjuntos de claves que poder rotar cuando se alcance el límite de peticiones.

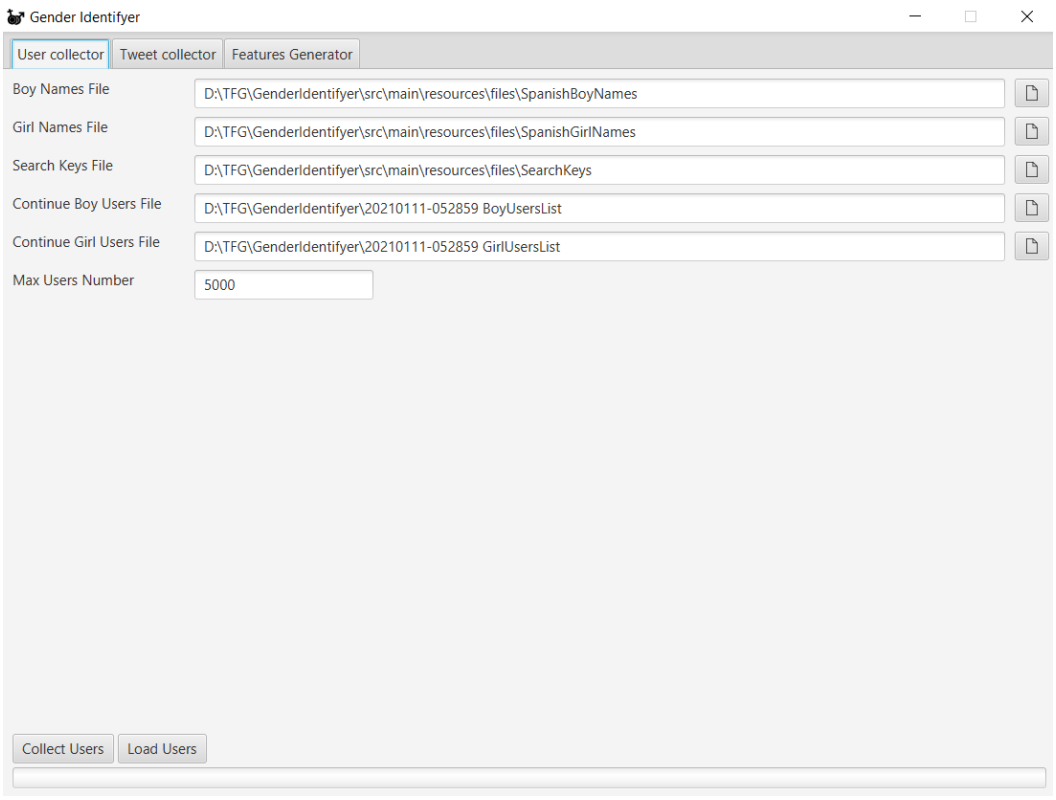
Recolección de usuarios

Para iniciar la aplicación será necesario contar con una carpeta llamada properties que contendrá los ficheros:

- genderIdentifyer.properties
- genderIdentifyer1.properties (con las mismas claves que el anterior)
- genderIdentifyer2.properties (claves diferentes)
- genderIdentifyer3.properties (claves diferentes.

Todas las trazas de la misma se guardan en la carpeta de logs.

El número de usuarios deseados es configurable y se buscan tantos usuarios hombres como mujeres.



En primer lugar, habrá que guardar las claves de Twitter en el fichero genderIdentifyer.properties.

Para realizar la búsqueda es necesario proporcionar a la aplicación un fichero de palabras a buscar (*Search Keys File*). Este fichero debe contener una palabra al menos (pueden ser varias) por línea, en español y preferiblemente de temáticas diferentes. El objetivo es realizar búsquedas generales en el timeline de Twitter de esas palabras y extraer los usuarios de los tuits publicados que la contengan

También se debe dar la ubicación de un fichero con nombres exclusivamente de hombres y otro con nombres exclusivamente de mujeres (*Boy Names File, Girl Names File*). Los nombres contenidos en estos ficheros deberían ser en principio identificables exclusivamente con el género masculino o con el femenino, por lo que si un usuario tiene uno de esos nombres se le atribuirá el género que corresponda a ese nombre.

Los archivos *Continue Boy Users File* y *Continue Girl Users File* contendrán los usuarios guardados previamente, si no se establecen se crearán unos nuevos con el siguiente formato: *[año][mes[día]-[horas][minutos][segundos]* seguido de *BoyUsersFile* o *GirlUsersFile*, según corresponda.

El botón *Collect Users* iniciará la recolección de usuarios, continuando a partir de los archivos de usuarios proporcionados o desde cero si no se han seleccionado.

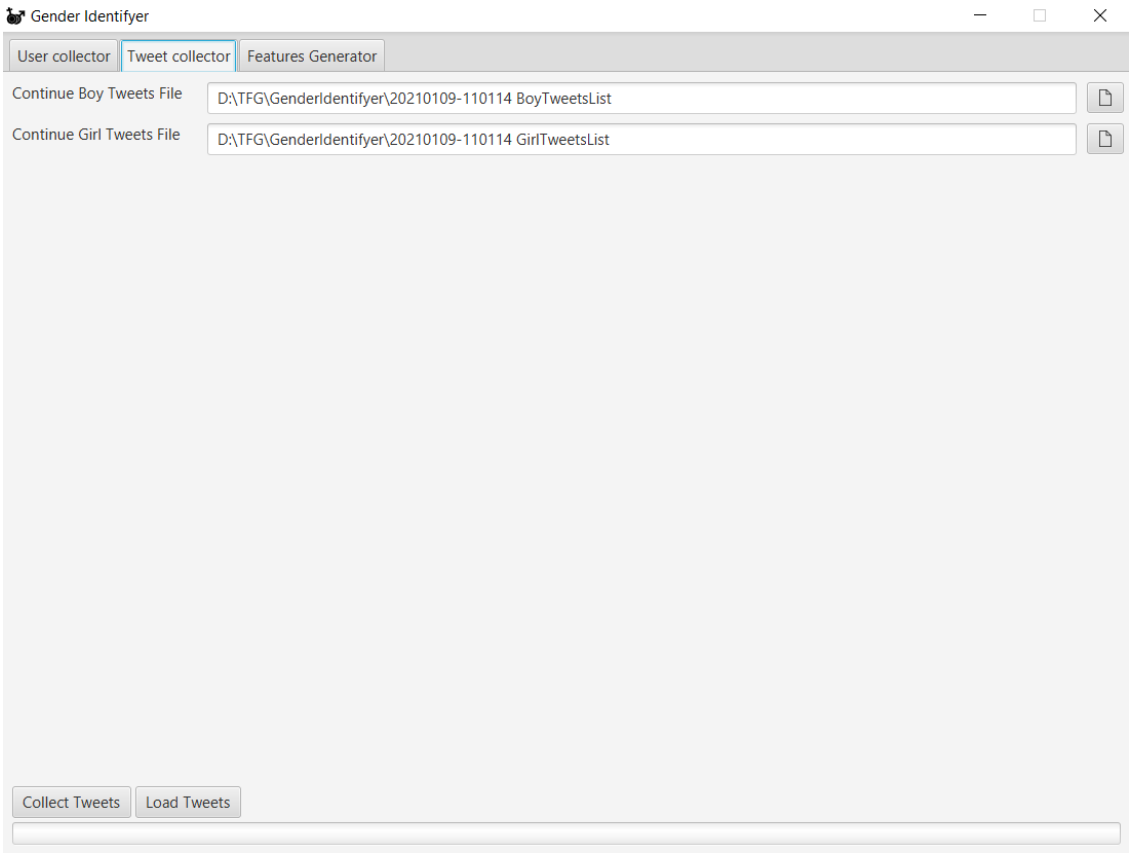
Por otra parte, el botón *Load Users* únicamente carga en la aplicación las listas de usuarios contenidas en los archivos de usuarios que se deben haber seleccionado previamente.

La ejecución finalizará con la extracción de todos los usuarios o un *RateLimit* que se puede consultar en el fichero de log. Tras ello es mejor opción seguir con la recolección de tuits y tras ella recolectar los usuarios restantes.

Recolección de mensajes (tuits)

Con las listas de usuarios ya creadas, es viable comenzar la recolección de los tuits de los mismos. Es necesario haber recolectado justamente antes los usuarios o haberlos cargado previamente.

También deberán existir los ficheros genderIdentifyer1.properties, genderIdentifyer2.properties y genderIdentifyer3.properties, que tendrán conjuntos de claves diferentes para rotar al alcanzar el limite de peticiones.



Muy similar a la pestaña anterior, los archivos *Continue Boy Tweets File* y *Continue Girl Tweets File* contendrán los tweets que se hayan recolectado previamente. Si no se seleccionan se generarán unos nuevos con el siguiente formato: *[año][mes[día]-[horas][minutos][segundos]* seguido de *BoyUsersFile* o *GirlUsersFile*, según corresponda.

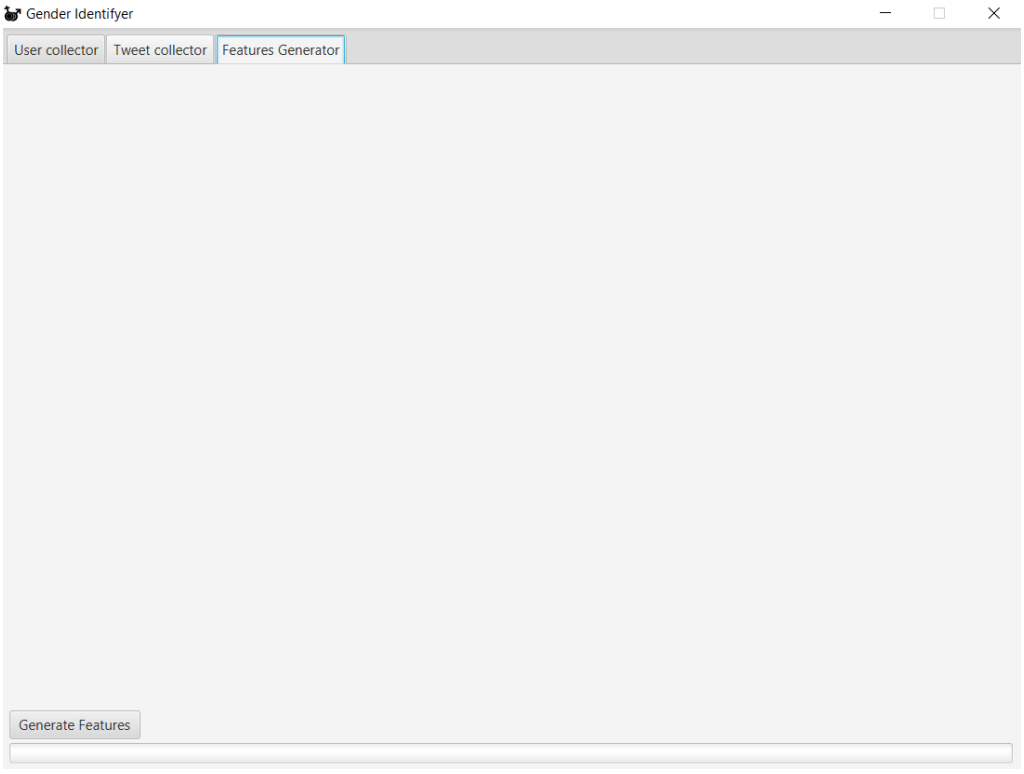
También existen dos botones con funciones análogas a las de la otra pestaña.

El botón *Collect Tweets* iniciará la recolección de los tuits de los usuarios que se hayan cargado en el sistema. Continuará a partir de los archivos de tuits si estos se le han proporcionado o si no comenzará de cero

La función de *Load Users* únicamente es la de cargar los tweets que estén guardados previamente en los dos archivos de tuits que se le pueden proporcionar a la aplicación.

Procesamiento del texto y extracción de características

Para poder comenzar con el procesamiento del texto es necesario haber recolectado o cargado en primer lugar las listas de usuarios y posteriormente haber recolectado o cargado los tuits de los mismos.



Es la pestaña más sencilla, únicamente existe un botón *Generate Features* que es el encargado de iniciar el procesamiento de los tuits cargados en la aplicación.

Para cuantificar estas características se procesan usuarios hombres y mujeres y por cada usuario se recorren todos los tuits.