UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DETECCIÓN AUTOMÁTICA DE SEGUIDORES FALSOS EN TWITTER

Álvaro Prieto Blanco
Tutor: Álvaro Ortigosa Juárez

JUNIO 2018

Detección automática de seguidores falsos en Twitter

AUTOR: Álvaro Prieto Blanco
TUTOR: Álvaro Ortigosa Juárez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2018

Resumen

La existencia de Redes Sociales en la actualidad conlleva, aunque en la mayoría de los casos es bueno y de gran utilidad hacia la comunidad que las consume, a veces un uso inadecuado de las mismas y, en numerosas ocasiones, ilegal.

Conseguir la ‘fama’ en estas comunidades, ya sea para fines publicitarios, económicos o de simple entretenimiento, parece el objetivo de muchas personas o colectivos que tratan por todos los medios de obtener el mayor número de visitantes, visores o seguidores en sus perfiles públicos.

Aquí viene el tema principal; a los perfiles que intentan conseguir esos seguidores, que tanta importancia mediática implican, no les resulta una tarea sencilla en muchos casos y recurren a la contratación de seguidores falsos.

Un seguidor falso no es más que una cuenta automatizada tecnológicamente que intenta imitar el comportamiento de un usuario real.

A partir de esta situación ilegal surge la necesidad, por parte del sector tecnológico, de desarrollar herramientas que la prevengan, o en su defecto la detecten y así tomar las medidas adecuadas.

Esta es la parte que corresponde a este Trabajo Fin de Grado. Consta del desarrollo de una herramienta que detecte este tipo de comportamiento de usuarios, que serán denominados ‘bots’.

La herramienta es realizada sobre una de las mayores y más utilizadas redes sociales, Twitter: una comunidad en la cual los usuarios publican pequeños posts conocidos como ‘tweets’ hacia otros usuarios en una relación de seguimiento entre perfiles. Los usuarios pueden interactuar con otros usuarios mediante dichos tweets.

Este trabajo consiste en el trato de una serie de tweets y usuarios reales y falsos almacenados en una base de datos no relacional para poder extraer datos relevantes a partir de ellos. Estos datos son obtenidos a partir de la API de Twitter y procesados mediante algoritmos en el lenguaje de programación Python.

El correcto formato, trato de los datos y ayuda de las librerías de Python enfocadas a machine learning, ha permitido formar un modelo de entrenamiento óptimo con el cual detectar si los usuarios de Twitter corresponden a personas reales, o cuentas automatizadas.

Palabras clave

Red social, Twitter, tuit, seguidor, bot, aprendizaje automático, API, modelo, algoritmo, conjunto de entrenamiento, conjunto de test, Base de Datos.

Abstract

The existence of Social Networks currently entails, although in most cases it is good and very useful for the community that consumes them, sometimes an inappropriate use of them and, in many cases, ilegal.

To achieve the 'fame' in these communities, whether for advertising, economic or simple entertainment purposes, it seems the goal of many people or groups that try at any cost to obtain the largest number of visitors, viewers or followers in their public profiles.

Here comes the main theme; to the profiles that try to obtain those followers, that so much mediatic importance implies, do not find it an easy task in many cases and they resort to the hiring of fake followers.

A fake follower is nothing more than a technologically automated account that attempts to mimic the behavior of a real user.

From this illegal situation arises the need, on the part of the technological sector, to develop tools that prevent it, or failing to detect it and thus take the appropriate performance.

This is the part that corresponds to this Final Degree Project. It consists of the development of a tool that detects this type of user behavior, which will be called 'bots'.

The tool is made on one of the largest and most used social networks, Twitter: a community in which users publish small posts known as 'tweets' to other users in a relationship between profiles. Users can interact with other users through these tweets.

This work consists in the treatment of a serie of tweets and real and fake users stored in a non-relational database in order to extract relevant data from them. This data is obtained from the Twitter API and processed by algorithms in the Python programming language.

The correct format, treatment of the data and by help of the Python libraries focused on machine learning, has allowed to perform an optimal training model with which to detect if Twitter users correspond to real people, or automated accounts.

Keywords

Social network, Twitter, tweet, follower, fake follower, machine learning, API, model, algorithm, training set, test set, Data Base.

Agradecimientos

Quiero agradecer a mi familia por haber permitido formarme iniciando mis estudios en la carrera y estar hoy en este punto de mi vida. Remontándose también al ámbito personal por hacerme crecer como persona y poder ahora seguir hacia delante con un poco más de independencia.

Dentro de este grupo incluyo a mis amigos de la infancia y actuales con los que comparto momentos increíbles y me permiten esos necesarios instantes de desconexión.

También agradecer a mi tutor de este proyecto, Álvaro Ortigosa, por haber aceptado la propuesta que le hice en su momento y haberme guiado en el desarrollo del trabajo.

Gracias a la Universidad Autónoma de Madrid por haberme acogido en mis estudios, en especial a la Escuela Politécnica Superior y a sus profesores.

Finalmente agradecer a mi entorno profesional, que desde hace unos meses me han posibilitado empezar una nueva etapa para seguir creciendo y formándome de aquí en adelante.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

No cabe ninguna duda sobre el abundante uso de las redes sociales virtuales[1] en la actualidad. Desde hace unos años hasta ahora y, posiblemente en el futuro, las tecnologías sigan creciendo, y con ellas, las redes sociales. Casi todas las personas a las que se le pregunte por la calle sobre si usan algún tipo de red social responderán afirmativamente.

Pero no todo el mundo es consciente de lo que hay detrás de las redes sociales y, es que en muchas ocasiones no se hace un correcto uso de estas. Por eso es importante, con el crecimiento de esta tecnología, un mayor cuidado para los usuarios que las consumen.

Concretamente se debe controlar y eliminar el uso de cuentas automatizadas cuyo contenido no sea el más adecuado, como puede ser publicar spam, seguir cuentas automáticamente o interactuar con publicaciones haciendo 'me gusta'.

Muchas veces, el éxito de una persona o de una marca se mide por la importancia que tiene en redes sociales y dos de los factores más significativos para medirla es el número de seguidores que tiene en sus perfiles públicos o por el éxito que tienen sus publicaciones expresado en cantidad de 'me gusta' que obtienen en ellas. Si dicha cantidad es muy grande puede generar una impresión de 'calidad' cuando en realidad no lo es si lo ha conseguido mediante interacciones automáticas.

Como consecuencia a este interés, existen empresas que ofrecen servicio a los usuarios vendiéndoles seguidores falsos por un módico precio. Esta práctica es una tentación para aquellos usuarios que buscan el reconocimiento en la sociedad con el fin de aumentar el prestigio social de su marca o recibir beneficio económico de esta en función del número de personas con las que pueden mediar. Una cuenta con millones de seguidores incita a otros usuarios a seguirla y, con esto, generar un mayor interés en el resto de los usuarios.

La creación de estos perfiles falsos puede darse a partir de usuarios inactivos cuyos perfiles son 'clonados' suplantando así su identidad con su nombre, foto, datos personales… haciendo creer que se trata de una persona real.

Se debe poder lidiar con este tipo de situaciones ilegales que se dan en las redes sociales.

1.2 Objetivos

El fin principal de este trabajo es el desarrollo de una herramienta que permita analizar los perfiles de los usuarios de Twitter con el objetivo de clasificarlos en cuentas automatizadas (bots) o cuentas reales (humanos).

Este proceso de análisis y clasificación se realizará utilizando métodos de aprendizaje automático supervisado.

[1] Por brevedad, a partir de ahora, el término "redes sociales virtuales" se denotará como "redes sociales".

El primer objetivo que se ha planteado es el estudio y análisis de los usuarios de Twitter y sus tweets. Se intenta determinar patrones que pueden seguir las cuentas automatizadas.

Con ello lo que se pretende obtener es una serie de atributos que son clave en el comportamiento de los usuarios en relación con los patrones identificados.

El segundo objetivo es la clasificación de usuarios en 'bot' o 'humano' partiendo de una base de datos de perfiles de Twitter etiquetados como tal, junto con sus últimos tweets.

1.3 Conceptos

- Machine learning:

  Rama de la IA que pretende desarrollar métodos para identificar patrones complejos a partir de datos.

- Reconocimiento de patrones:

  Tiene el objetivo de identificar una serie de comportamientos sobre un objeto, persona, o actividad.

- Característica, atributo o factor:

  Definen y describen a un objeto, persona, o actividad y, en su conjunto, lo clasifican.

- Aprendizaje supervisado:

  Técnica del aprendizaje automático que se basa en la relación de los datos que se tienen almacenados y el campo objetivo para hacer predicciones futuras sobre ese campo en función de las características de entrada.

- Aprendizaje no supervisado:

  Esta técnica, por el contrario, no utiliza características etiquetadas en un principio, sino que busca una manera de agrupar esos datos. No hay un conocimiento 'a priori'.

- Dataset:

  Conjunto de datos con los que se va a trabajar para el entrenamiento. Consta de varios registros y, cada uno de ellos, de características o atributos.

- Entrenamiento:

  Es el proceso por el cual se utiliza un dataset sobre un algoritmo para generar un modelo.

- Modelo:

  Resultado generado a partir del entrenamiento que se usa sobre unos datos nuevos con el fin de predecir una salida para clasificar la entrada.

1.4 Organización de la memoria

La memoria consta del siguiente esqueleto:

- Estado del arte tanto de las redes sociales como del aprendizaje automático y tecnologías a utilizar.
- Diseño de la herramienta del trabajo, con requisitos, trabajo previo y arquitectura.
- Desarrollo del trabajo.
- Pruebas y resultados del trabajo.
- Conclusiones y trabajo futuro.

2 Estado del arte

2.1 Redes sociales

Nadie duda del impacto que han tenido las redes sociales en los últimos años que involucran a millones de usuarios a nivel mundial.

Cualquier marca que quiera ser competente en el mercado debe tener algún tipo de red social por la cual se dé a conocer y sirva de mecanismo para el negocio.

2.1.1 Historia

Las redes sociales como las conocemos actualmente y aplicadas al entorno digital surgieron en el año 1995 con la aparición de internet en nuestras vidas.

En 1995 surgió Classmates [1], cuyo propósito era el de encontrar amigos de estudio. Posteriormente, en 1997, surgió Sixdegrees [2], la primera en tener perfiles, con listas de amigos y la inclusión de los mensajes como medios de comunicación.

Entrando en el año 2000 iban surgiendo redes orientadas a la intercomunicación de usuarios. Las más conocidas fueron LinkedIn [3] (2002), de contactos profesionales y Facebook [4] (2004) más bien enfocada al entretenimiento. Estas dos siguen vigentes en la actualidad y son las más populares en su objetivo.

En 2006 se lanza Twitter [5], la red de 'microblogging' conocida mundialmente, y nacen redes de contenido multimedia como Instagram [6] (2010) o Snapchat [7] (2011).

2.1.2 Usuarios en las Redes Sociales

Como he comentado antes, el impacto de las redes sociales viene dado por la cantidad de usuarios que son capaces de mover.

Las tres redes sociales más competentes actualmente son Facebook, Twitter e Instagram.

A fecha de Julio de 2017, se registran un total de 23 millones de usuarios de Facebook en España, lo que equivale a un 50% de la población, y 2070 millones de usuarios a nivel mundial, más de 800 millones de usuarios en Instagram, y 330 en Twitter [8].

La evolución en España en los 4 últimos años de estas redes sociales se muestra en la Figura 1.

	2014	2015	2016	2017
Facebook	20 millones	22 millones	24 millones	23 millones
Twitter	3,5 millones	4,4 millones	4,5 millones	4,9 millones
Instagram		7,4 millones	9,6 millones	13 millones

Figura 1. Evolución del número de usuarios en Facebook, Twitter e Instagram en España [8].

A nivel mundial y en términos generales, en tan solo 6 años casi se ha duplicado el numero de usuarios que usan redes sociales. La figura 2 muestra el evolutivo [9].

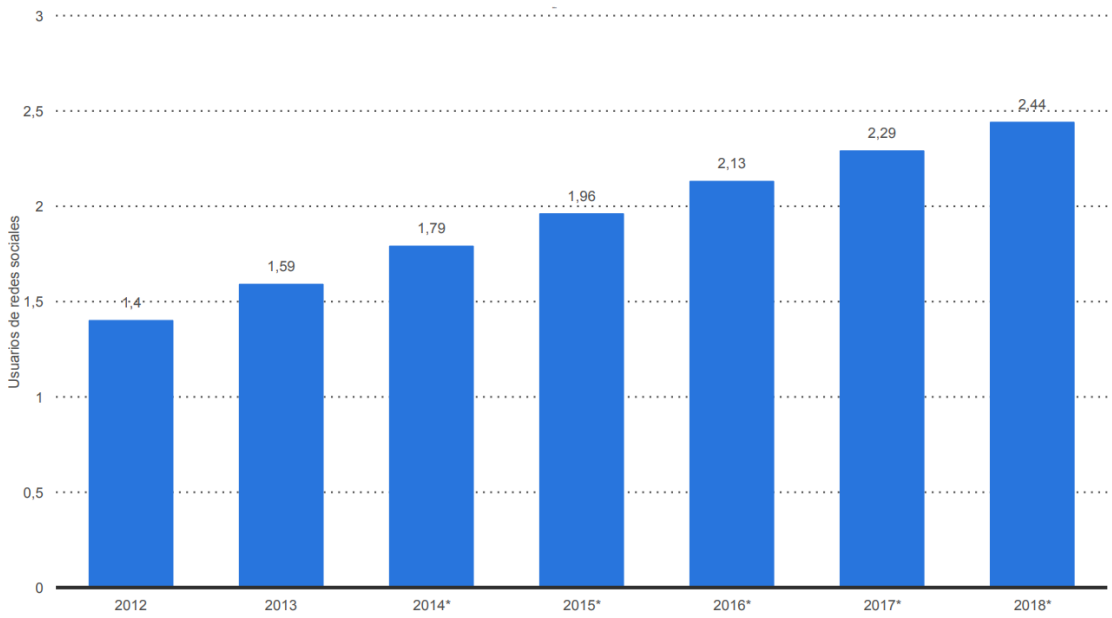

Figura 2. Evolutivo del uso de redes sociales 2012-2018 a nivel mundial [9].

Asimismo, es interesante estudiar el consumo diario de los usuarios. La figura 3 muestra como incrementa con el paso de los años [9].

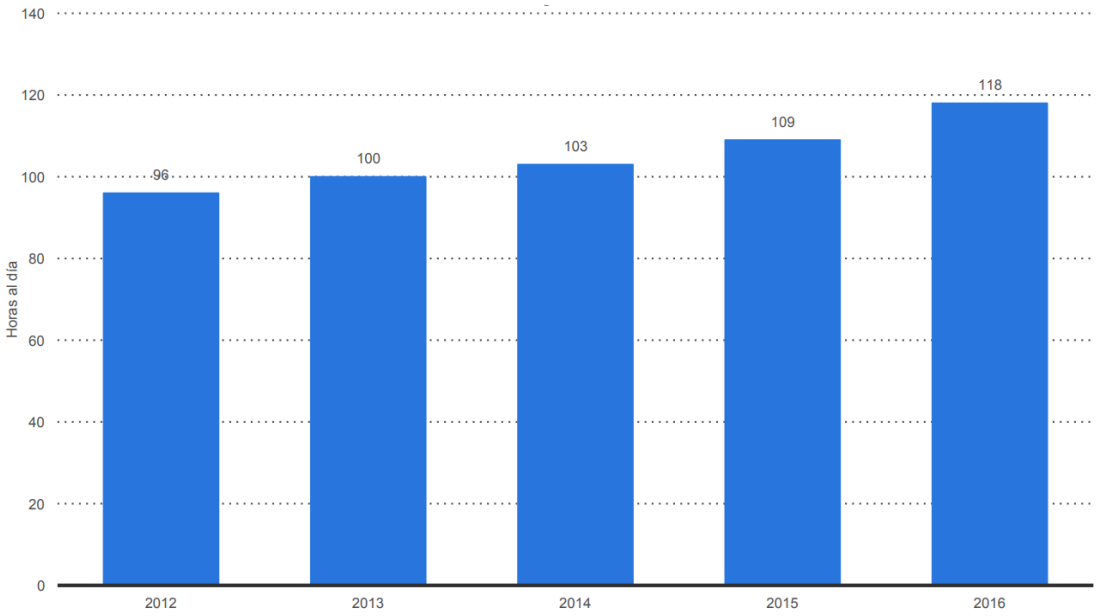

Figura 3. Evolutivo de uso diario de redes sociales. 2012-2018 (En minutos) [9]**.**

2.1.3 Twitter

Uno de los puntos fuertes de Twitter es la información transmitida en tiempo real, además de ser un medio de comunicación para las marcas y compañías.

Su auge fue durante el periodo de 2013 a 2015 llegando desde los 600 millones de usuarios a los 2500 en tan solo 3 años. En 2015 sufrió un estancamiento debido al aumento de popularidad de otra gran red social, Instagram (Figura 6). Empezó el año con el mismo número de usuarios que al terminarlo. Actualmente tiene un crecimiento lento con respecto al del resto de redes sociales [9]. Esta situación se muestra en las figuras 4 y 5 con los ingresos que ha obtenido y la actividad de usuarios por trimestre que ha adquirido.

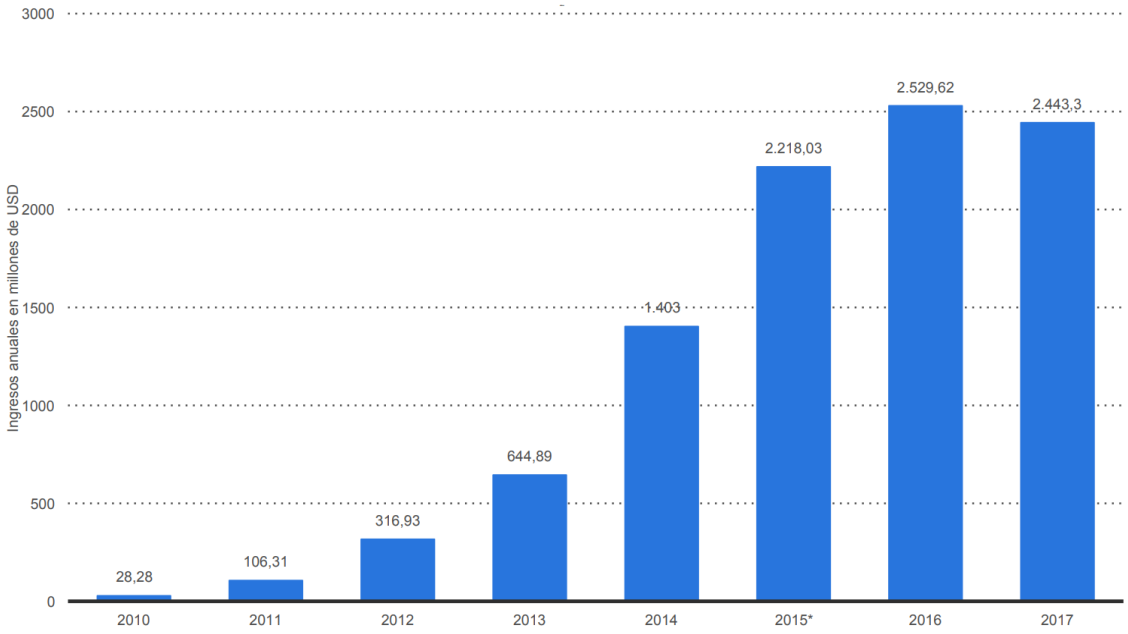

Figura 4. Ingresos anuales 2010-2017.

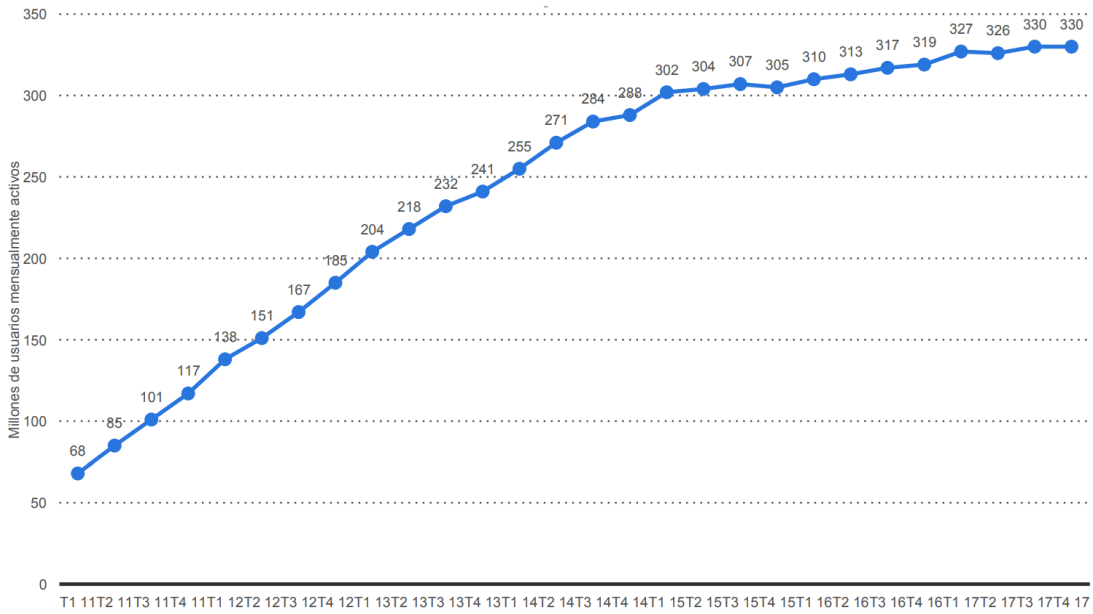

Figura 5. Usuarios mensuales activos por trimestre 2011-2017.

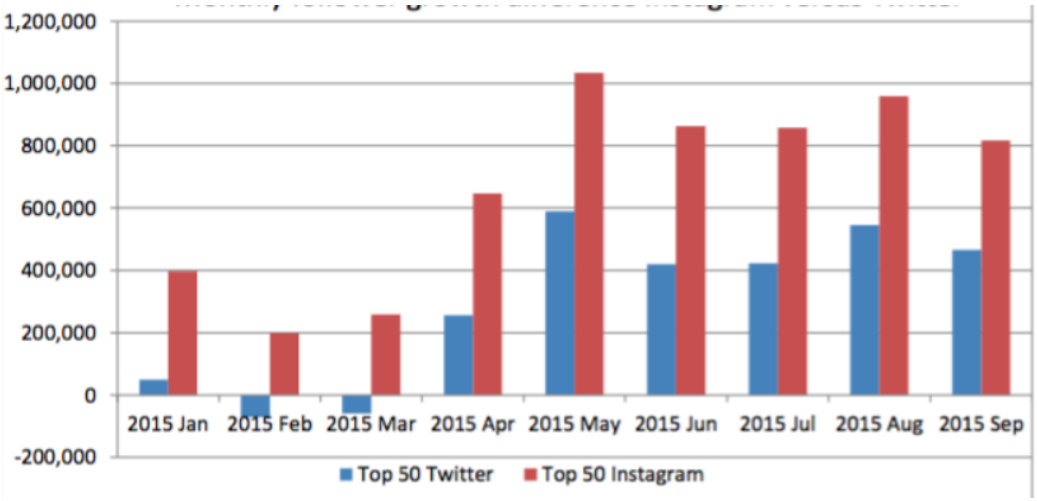

Figura 6. Diferencia de crecimiento en cuanto a usuarios de Twitter vs Instagram.

2.1.4 Perfiles automatizados

Con la importancia de las redes sociales en la comunicación e información, se ha recurrido al uso de 'bots' o cuentas automatizadas para llevar a cabo distintas funciones cuyo objetivo es publicitario, económico o político.

2.1.4.1 Perfiles automatizados en Twitter

Concretamente Twitter dispone de un API para desarrolladores que puede ser usado para obtener información y enviar peticiones para por ejemplo publicar, dar 'favoritos' a los tweets de manera automática o seguir a otras cuentas de Twitter. Debido a ello surgen las cuentas automatizadas, o más conocidas como 'bots'.

Se estima que, en Twitter, el 52% del tráfico es por parte de 'bots' a fecha de 2016 [15].

¿Cómo actuamos ante estas situaciones? Lo primero es conocer como identificarlos buscando etiquetas que los caracterizan [13].

A simple vista podemos intuir algunas de ellas que parecen más que evidentes, por ejemplo, retweets masivos en un corto espacio de tiempo, seguir a muchas cuentas sin tener muchos seguidores, no tener imagen principal o tener la que Twitter establece por defecto, Twittear desde ciertas API's, etc.

Pero estas cuentas van mucho más allá. No basta con las características que podemos intuir, sino que se debe hacer un estudio más en profundidad sobre ellas [14].

Para este trabajo se han identificado una serie de características mostradas en el apartado 3.3.

Como expone Xakata en su artículo [14], tanto alguna Universidad como la propia empresa de Twitter han intentado crear herramientas que detecten estas cuentas. Sin embargo, no han obtenido un gran éxito. No todos los usuarios que se identifican con una serie de características propias de 'bots' tienen que ser 'bots'. Por ejemplo, una de las más

identificativas para un ‘bot’ es la constante publicación de tweets. Sin embargo, existen muchos usuarios reales que también twittean de forma consecutiva.

De igual manera, se han desarrollado herramientas que analizan el porcentaje de seguidores ‘bot’ y quienes son, como es el caso de Botometer [25] (antiguamente llamada BotOrNot), desarrollada en Python por la Universidad de Indiana Bloomington, y cuyo estudio se ve en [13]. Utiliza Pandas [26], NLTK [27] y scikit-learn [16] para identificar estos ‘bots’.

Otra herramienta es BotCheck [28], desarrollada por dos estudiantes, Ash Bhat y Rohan Phadte, que utiliza también aprendizaje automático para, en este caso, detectar ‘bots’ de propaganda política. Se puede ver un análisis de esta en [29].

2.1.4.2 Estudio de los atributos de un perfil automatizado en Twitter

La evaluación de los atributos o características clave para poder detectar patrones en el comportamiento de los usuarios de Twitter es fundamental si queremos obtener los mejores resultados posibles en la precisión y predicción de nuevos usuarios.

Gracias a las investigaciones de Zafar Gilani en [15] y [23] junto a Ekaterina Kochmar y Jon Crowcroft, Chao Michael Zhang, Vern Paxson en [22] y otros autores en [21] se han podido estimar caracterizaciones de los usuarios y estudiar los atributos que pueden diferenciar a un ‘bot’ de un ‘humano’. Son los siguientes:

- La ‘edad’ de la cuenta: Los ‘bots’ tienden a ser más nuevos y renovarse, mientras que los humanos llegan a tener cuentas más viejas.
- Número de favoritos: Los ‘bots’, con la finalidad de conseguir presencia en la comunidad, tienden a provocar reacciones en otros usuarios etiquetando los tweets de estos como favoritos.
- Listas a las que están suscritos: Esta puede ser una característica importante. Los ‘bots’ pueden suscribirse a listas de Twitter donde hay más usuarios a los que pueden seguir. Además, hay listas por temática, y los ‘bots’ pueden elegir preferencias en estas.
- Ratio friends/followers: Esto es algo que caracteriza típicamente a los humanos. Este ratio para ellos suele ser próximo a 1, teniendo el mismo número seguidores que amigos, aunque existe gente ‘famosa’ que tiene más seguidores que amigos y puede dar lugar a falsos positivos en la clasificación. En caso de los ‘bots’, este ratio tiende a ser mucho más alto, teniendo más amigos que seguidores.
- Ratio retweets/tweets: Los ‘bots’ tienden a retweetear más que a publicar tweets. Es la forma más simple de causar una reacción en un usuario.
- Origen desde donde tweetean: Los ‘bots’ suelen usar herramientas o API’s que les ayuden a realizar su trabajo mientras que los humanos tienden a usar Twitter desde el móvil o desde la Web.
- Número de tweets: Esta característica puede ser confusa. En principio, un ‘bot’ puede llegar a tener un número alto de tweets, sin embargo, hay usuarios que pueden llegar a igualarlo.

- Frecuencia de tweeteo: De la misma manera, un ‘bot’ tiende a ser constante en sus interacciones publicando tweets o retweeteando, por lo que puede ser característico el tiempo que pasa entre un tweet y otro.
- Respuestas a usuarios: Otro tipo de interacciones que pueden ser características en una cuenta automatizada es, constantemente, contestar a otros usuarios. Por ello se ha medido el número de respuestas en sus tweets.
- Número de hashtags y URL’s: Los ‘bots’ tienden a tener hastaghs en sus tweets para etiquetarlos con respecto a algún tema y el resto de usuarios puedan verlos. De igual manera pasa con las URL’s. Hay muchos ‘bots’ con enlaces de publicidad o spam.
- Número de menciones a usuarios: En sus tweets los ‘bots’ tienden a mencionar a los usuarios.
- Tiempo entre el tweet y retweet: Existen ‘bots’ encargados de ‘cazar’ tweets con etiquetas e interaccionar con ellos. Es por esto que esta puede ser una característica importante. Al momento que un ‘bot’ detecta un tweet que quiere, lo retweetea.

2.2 Aprendizaje automático

Inicialmente la IA se basaba en el desarrollo de sistemas expertos, capaces de emular las decisiones que toma un experto humano en la resolución de un problema. Además, también participaba un ingeniero del conocimiento, que extrae reglas a partir de entrevistas con el experto y las utiliza para crear el programa. Esta interacción es el problema principal en el desarrollo de la aplicación, que se podría mejorar automatizando el proceso de ingeniería del conocimiento o consiguiendo que el sistema pueda aprender a ser un experto a base de ejemplos.

Lo que se busca en un sistema es darle la capacidad de adaptarse sin ser reprogramados. A parte existe otro gran problema y es que la mayor parte de sistemas de Inteligencia Artificial son tan grandes que no pueden construirse directamente y es necesario que tengan algún tipo de algoritmo que permita incorporar información nueva de forma automática [10]. Eso es lo que se conoce como “entrenar un algoritmo”. Como resultado a ese entrenamiento tenemos un modelo.

Existen dos grandes áreas principales o técnicas en las que se divide el aprendizaje automático según la forma de entrenar el algoritmo: supervisado y no supervisado.

El primer grupo parte de un dataset de entrenamiento de entrada clasificado con la etiqueta correcta para intentar clasificar un segundo dataset sin etiqueta. Cuanto más grande pueda ser el conjunto, más podrá aprender el algoritmo. En los no supervisados no se dispone de un conjunto de entrenamiento previo, sino que a partir del dataset de entrada que nos dan, se analizan sus propiedades para encontrar algún tipo de patrón e intentar agrupar (clasificación o clustering) los registros del dataset según su semejanza.

2.3 Tecnologías empleadas

2.3.1 API de Twitter

Para acceder a los métodos de la API REST de Twitter existen varias librerias.

La que se ha utilizado en este trabajo es Twython [12] debido a que está en constante actualización y da soporte para Python 2.7 de manera sencilla.

2.3.2 Pyhton y librería Python scikit-learn

Python es un lenguaje con gran potencial para el "machine learning". A veces se necesitan procesar grandes cantidades de datos, o incluso hacer retroceder para probar otro tipo de estrategias.

La librería scikit-learn [16] es de las más utilizadas hoy en día para tratar en el ámbito del aprendizaje automático. Contiene funciones intuitivas y sencillas para realizar todo tipo de operaciones además de hacer el trabajo pesado para obtener los modelos de los algoritmos de clasificación.

2.3.3 Base de datos no relacional: MongoDB

Es importante tener en cuenta la rapidez que hace falta para procesar grandes cantidades de datos y tratar con ellos. El esquema en este tipo de bases de datos no está estrictamente definido, sino que se pueden insertar documentos de forma flexible sin que afecte a la consistencia de datos. Es por ello por lo que MongoDB [18] es una base de datos perfecta para el almacenamiento de usuarios y tweets que provienen de la API de Twitter ya que se presentan en formato JSON coincidiendo con el formato admitido de los registros de MongoDB.

Las consultas a esta base de datos son triviales pudiendo filtrar por la 'key' del diccionario JSON que se desee.

2.3.4 Flask

Flask [17] es un framework para Python que permite crear aplicaciones web de forma sencilla siguiendo el patrón Modelo-Vista-Controlador (MVC). Por ello para mostrar los resultados de una manera no muy compleja, es el framework adecuado.

En el apartado 3.6 se muestran las pantallas que se han creado con Flask.

3 Análisis y Diseño

Una vez visto el estado del arte, se puede comenzar con el análisis y diseño del sistema.

Se va a proponer una solución software basada en un estudio de los requisitos que se han obtenido tras una entrevista con el cliente, que expone sus necesidades.

Tras este análisis, se propone un diseño de la arquitectura del sistema representando dichos requisitos y el flujo entre módulos o componentes de este.

Finalmente se expone el diseño de la interfaz de la herramienta.

3.1 Requisitos

3.1.1 Requisitos funcionales

RF 1. Utilización de varios modelos para la clasificación del usuario en ‘Bot’ o ‘Humano’.

3.1.2 Requisitos no funcionales

RNF 1. Utilización del lenguaje de programación Python.

RNF 2. Utilización de la librería “Twython” para la API de Twitter.

RNF 3. Utilización de la base de datos no relacional “MongoDB” para almacenar los datos.

RNF 4. Implementación de una interfaz simple para la utilización de la herramienta.

RNF 5. Comprobar la precisión del dataset cumpliendo un objetivo mínimo.

3.2 Arquitectura

Como muestra la figura 7, se ha seguido una arquitectura lo más modularizada y escalable posible, tipo estrella con varios componentes. Cada módulo tiene una funcionalidad específica de tal forma que se puedan hacer pruebas unitarias por separado en cada módulo y posteriormente pruebas de integración con todos ellos. Estas pruebas quedan reflejadas más concretamente en el apartado 5.

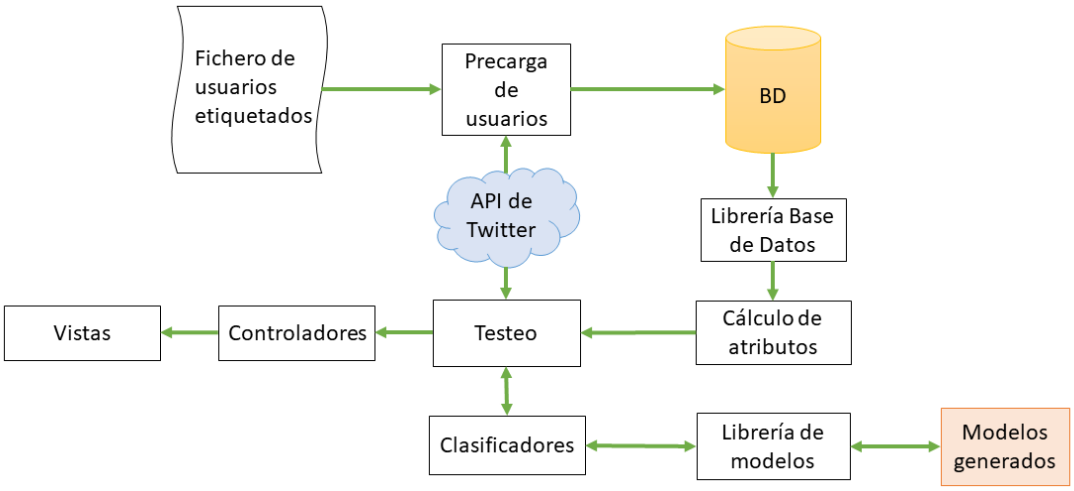

Figura 7. Arquitectura de componentes.

Módulo ‘Vistas’:

Contiene las pantallas de la interfaz correspondientes a la generación de módulos, testeo del dataset y predicción de usuarios. El controlador correspondiente del módulo ‘Controladores’ las renderiza.

Módulo ‘Controladores’:

Contiene los controladores para generar los modelos, testear el dataset y testear el usuario, y devolver una vista al usuario.

Módulo ‘Testeo’:

Módulo central, accedido por ‘Controladores’, que se encarga del manejo de llamadas al resto de módulos y a la API de Twitter para generar los modelos, testear el dataset y el usuario.

Módulo ‘Cálculo de atributos’:

Accedido por ‘test’, contiene la lógica para las llamadas a ‘Librería Base de Datos’, calcular los datos necesarios y devolverlos al módulo ‘Testeo’.

Módulo ‘Clasificadores’:

Accedido por ‘test’ para, una vez obtenidos los datos, calcular las precisiones y predicciones pertinentes o guardar los modelos y el dataset a través de ‘Librería de modelos’.

Módulo ‘Librería Base de Datos’:

Contiene la funcionalidad de llamadas a la Base de Datos para obtener los datos necesarios de usuarios y tweets y, posteriormente, ser devueltos y tratados en ‘Cálculo de atributos’.

Módulo ‘Precarga de usuarios’:

Encargado de parsear el fichero de entrada de ‘Fichero de usuarios etiquetados’ como ‘bot’ o ‘humano’ para obtener su información y tweets a través del API de Twitter y ser almacenados en la Base de Datos. Se utiliza manualmente y no forma parte de la herramienta en sí, sino de la configuración previa.

Módulo ‘Librería de modelos’:

Contiene las funciones para guardar y cargar los modelos y el dataset.

3.3 Diagrama de flujo

La figura 8 muestra el diagrama de flujo sobre la arquitectura correspondiente a la precarga de usuarios.

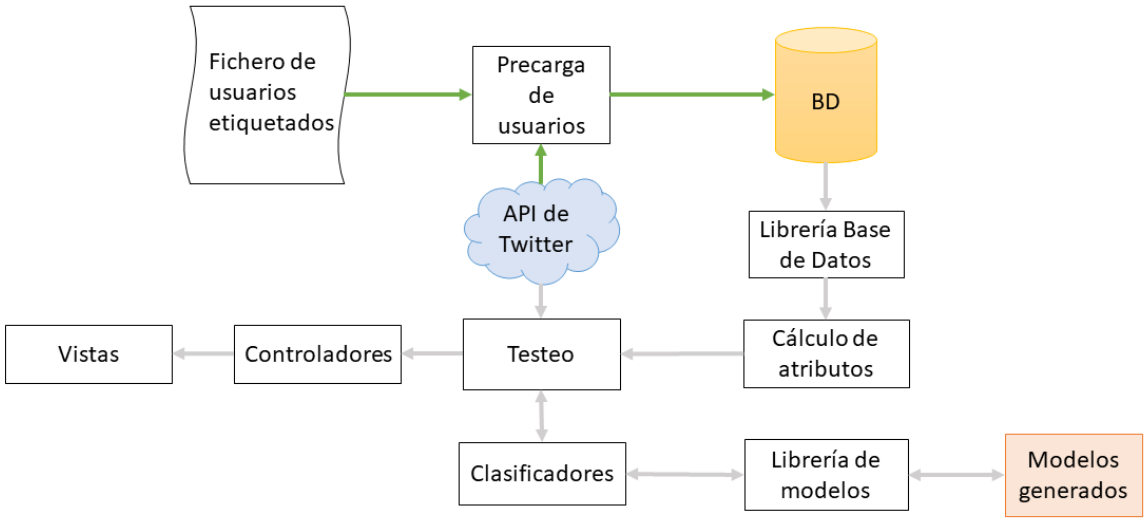

Figura 8. Diagrama de flujo sobre la arquitectura para la precarga de usuarios.

La parte correspondiente a la precarga de usuarios de Twitter se explica en la fase 1 del apartado 4 (desarrollo).

La figura 9 muestra el diagrama de flujo sobre la arquitectura correspondiente a la generación de los modelos.

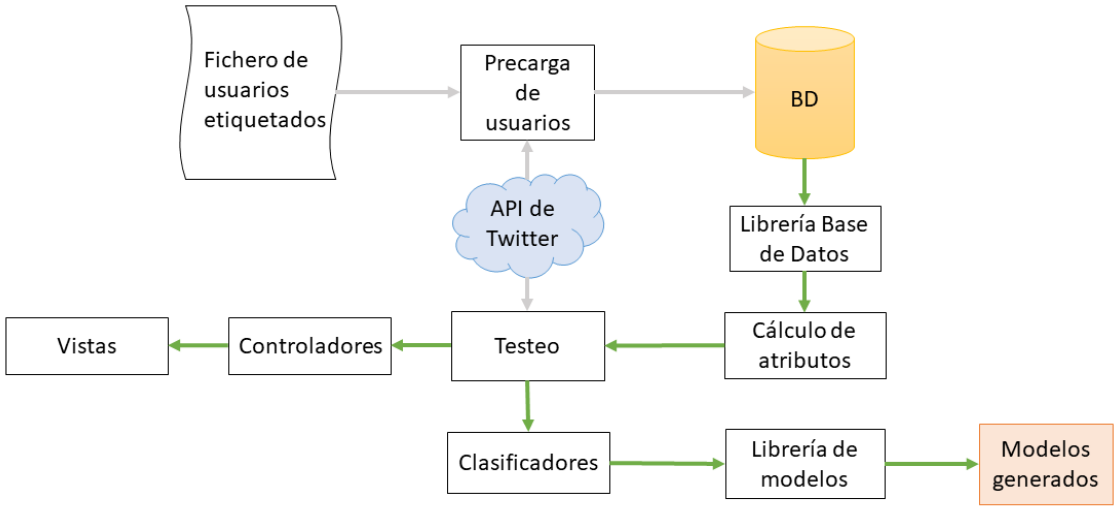

Figura 9. Diagrama de flujo sobre la arquitectura para 'generar modelos'.

Introduciendo el nombre de la base de datos a partir de la cual se quieren obtener los modelos se accede al módulo 'Controladores' para llamar a la función de 'generar_modelos'. Posteriormente, desde ahí, se accede al módulo 'Testeo', que es el que va a controlar el resto de la gestión de llamadas. En primer lugar, accede al módulo 'Cálculo de atributos' y este a su vez a 'Librería Base de Datos' para obtener los datos de usuarios y tweets. El módulo 'Cálculo de atributos' ejecuta la lógica necesaria en sus algoritmos para obtener, a partir de usuarios y tweets, todos los campos necesarios de entrenamiento. Este se los devuelve a 'Testeo' y hace una llamada a 'Clasificadores' donde todos los clasificadores llamaran a la función de 'save_model' para generar los modelos en una carpeta de 'Modelos generados' y así poder ser cargados posteriormente.

La figura 10 muestra el diagrama de flujo correspondiente al testeo del dataset guardado previamente.

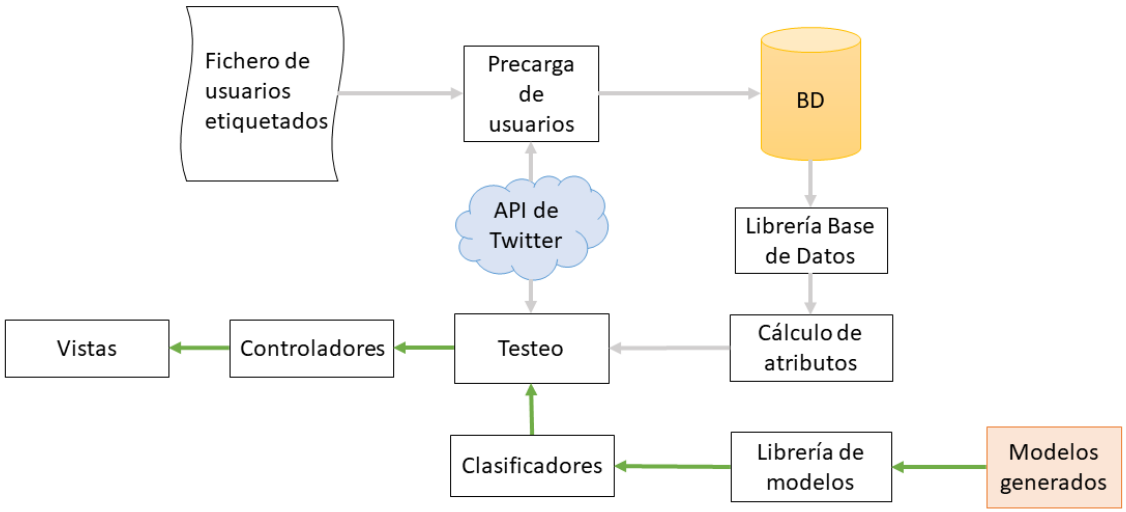

Figura 10. Diagrama de flujo sobre la arquitectura para 'testear dataset'.

Introduciendo el nombre del dataset y la proporción de datos de entrenamiento que se quiere testear del mismo, se accede al módulo 'Controladores' para llamar a la función de 'testear_dataset' del módulo 'Testeo'. Esta función lleva toda la lógica de llamadas al resto de los módulos. En este caso, solamente accede a 'Clasificadores' para que, a través de 'Librería de modelos' se obtenga el dataset guardado en la carpeta de 'Modelos generados'. Se devuelve al módulo 'Testeo' que dividirá el dataset en dos partes en función de la proporción introducida.

La figura 11 muestra el diagrama de flujo correspondiente al testeo del usuario introducido.

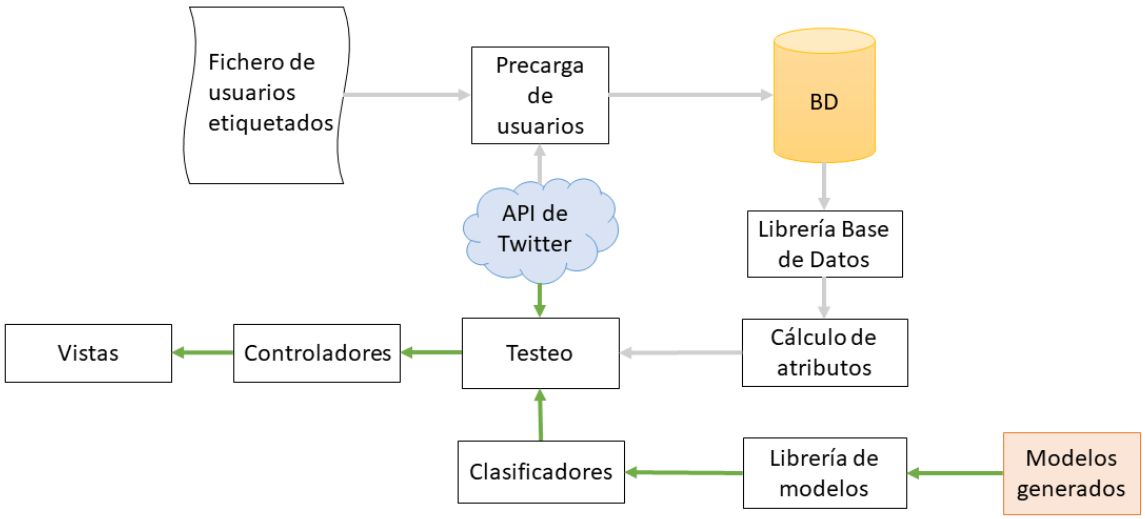

Figura 11. Diagrama de flujo sobre la arquitectura para 'testear usuario'.

Introduciendo el nombre del usuario de Twitter que se quiere testear, se accede al módulo 'Controladores' para llamar a la función de 'testear_usuario' del módulo 'Testeo'. Esta función lleva toda la lógica de llamadas al resto de los módulos. En primer lugar, se accede a la API de Twitter para obtener los datos del usuario y sus tweets. Posteriormente se accede a 'Clasificadores' donde se llamará a 'classifier_models', que tienen las funciones para obtener los modelos a devolver. Ya en el módulo 'Testeo', juntando ambos datos obtenidos se obtiene la salida correspondiente que aparecerá en la vista de la interfaz del usuario.

3.4 Interfaz de la herramienta

Para comprobar el funcionamiento de la herramienta se ha realizado una pequeña interfaz usando el microframework Flask, con ayuda de Bootstrap.

La figura 12 muestra una imagen de la pantalla principal con un formulario en el que se debe rellenar el usuario de Twitter con el que se desea testear los modelos o el nombre de la base de datos bien para testear su contenido con su proporción de entrenamiento o bien para generar los modelos.

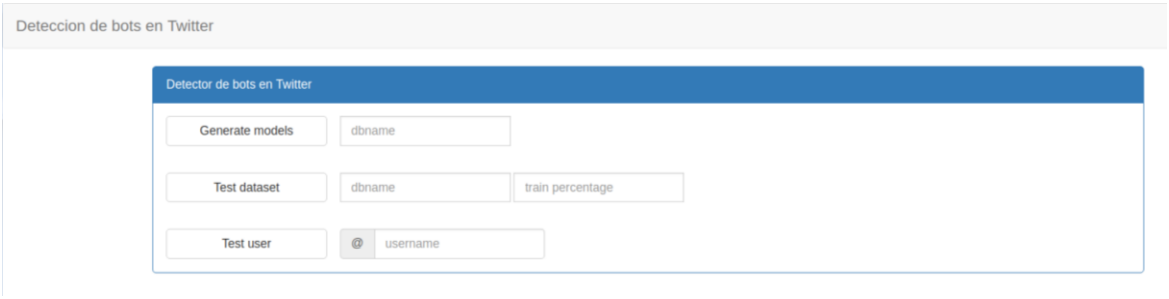

Figura 12. Formulario principal de la interfaz de la herramienta.

La figura 13 muestra el resultado de procesar el formulario tras pulsar ‘generar modelos’. Se guardan en la carpeta ‘Modelos’ los modelos generados y el dataset en formato csv.

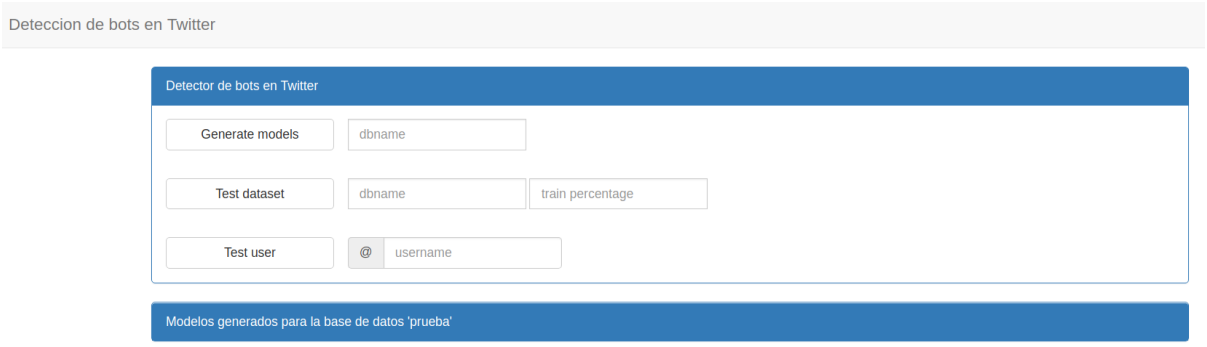

Figura 13. Resultados tras procesar el formulario con la petición de ‘generar modelos’.

La figura 14 muestra el resultado de procesar el formulario tras pulsar ‘testear dataset’. Se exponen las precisiones para cada uno de los algoritmos de clasificación entrenados.

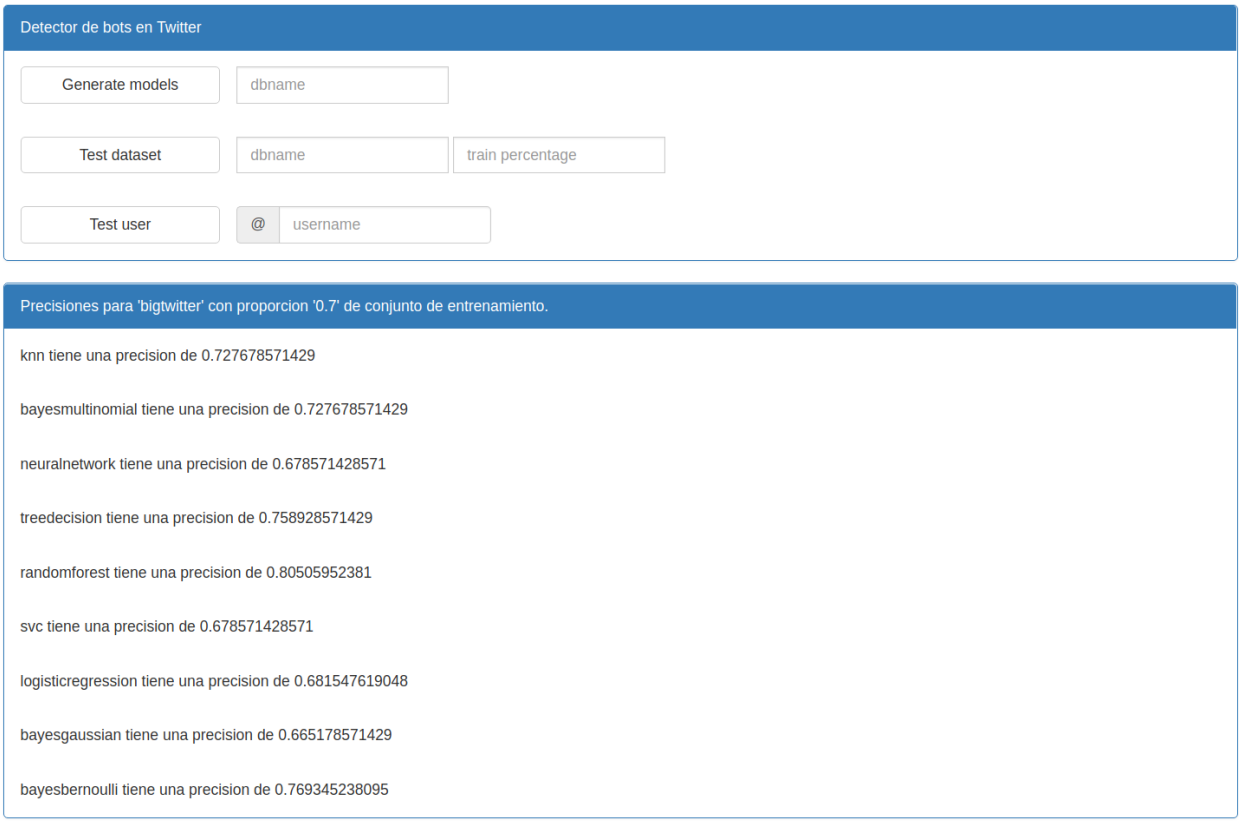

Figura 14. Resultados tras procesar el formulario con la petición de ‘testear dataset’.

La figura 15 muestra el resultado de procesar el formulario tras pulsar ‘testear usuario’.

Se exponen las predicciones obtenidas por cada uno de los algoritmos de clasificación entrenados.

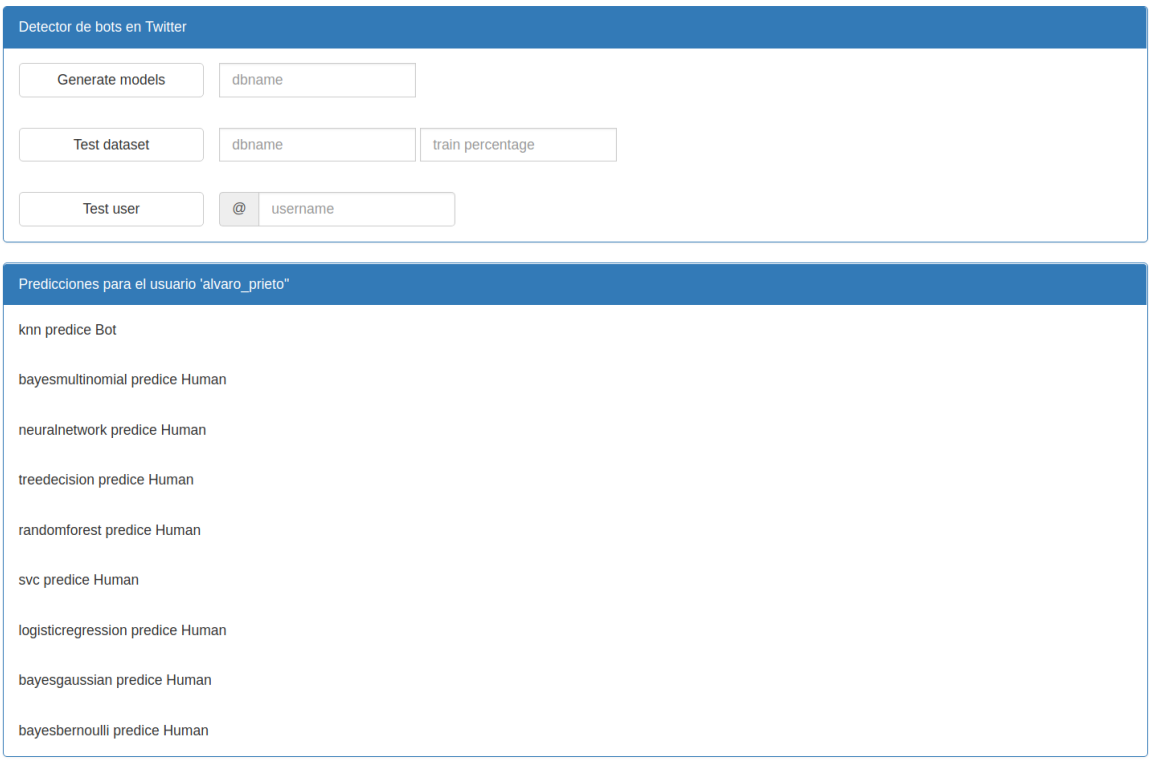

Figura 15. Resultados tras procesar el formulario con la petición de ‘testear usuario’.

4 Desarrollo

El desarrollo del trabajo se puede dividir en varias fases relacionadas a los módulos en los que están implicadas:

- **Fase 0**: Configuración.

  En esta fase se realiza la configuración pertinente para el funcionamiento de la herramienta.

  Primeramente, se ha creado una aplicación para desarrolladores en Twitter con la que poder acceder a la API en el código mediante los tokens y claves necesarios, generados por la aplicación y definidos en un fichero 'info.py'; OAUTH_TOKEN, OAUTH_SECRET, CONSUMER_KEY, CONSUMER_SECRET.

  Además, en este fichero se definen el número de tweets con la constante N_TWEETS que se quiere obtener por usuario. En nuestro caso 180.

  También se establece la configuración a la base de datos 'MongoDB'.

  MONGO_USER, MONGO_PWD, MONGO_HOST, MONGO_PORT definen el usuario, contraseña, host y puerto de la aplicación.

  StringConnection = "mongodb://" + MONGO_USER + ":" + MONGO_PWD + "@" + MONGO_HOST + ":" + MONGO_PORT + "/"

  StringConnection será la url con la que se abrirá conexión a la Base de Datos.

  APP_HOST y APP_PORT corresponden al host y puerto donde se lanzará la aplicación.

  En otro fichero 'info.py' se definen varias listas de atributos que se utilizarán para gestionar los datasets. Estas son 'input_attributes': atributos de entrada, 'encode_input_attributes': atributos de entrada codificados, 'output_attributes': atributos de salida, 'attributes': atributos totales.
  Y los atributos correspondientes a los objetos de twitter: 'attributesUser': campos de un usuario de Twitter y 'attributesTweet': campos de un tweet.

- **Fase 1**: Precarga de la Base de datos.

  Esta fase forma parte, en cierta medida, de la configuración previa en la que se almacenan los usuarios en la Base de Datos. Se ha conseguido una base de datos de usuarios de Twitter etiquetados como 'Bot' o 'Humano'. Gracias al repositorio de Botometer [21] se ha obtenido un fichero que contiene usuarios etiquetados como 'bot' o 'humano' junto con sus 'id's de Twitter.

  El fichero 'precarga.py' contiene dos métodos: 'parse_file' que limpia el fichero de usuarios con la cuenta protegida, no existentes o sin suficientes tweets para ser analizados. El otro método, 'read_from_file' lee el fichero y va procesando y obteniendo, fila por fila, los datos de los usuarios en formato JSON, según se

muestra en el API de Twitter [19] y sus últimos 180 tweets [20] e introduciéndolos en la tabla ‘users’ y ‘tweets’ de la Base de Datos.

- **Fase 2**: Creación de los métodos que llaman a la Base de Datos.

  En esta fase se implementa la librería de llamadas a la Base de Datos mediante métodos.

  El fichero ‘database.py’ contiene la clase DataBase con todos estos métodos.

  Esta clase contiene además los atributos ‘name_db’, nombre de la Base de Datos, ‘client’, cliente de conexión a MongoClient, ‘collectionUsers’ y ‘collectionTweets’, las colecciones correspondientes a los objetos ‘usuario’ y ‘tweet’ de la API de Twitter. Los métodos que se utilizan en el uso de la herramienta son ‘insert_user_db’ para insertar un usuario, ‘insert_tweet_db’ insertar un tweet, ‘get_user_by_id_from_db’ obtener el usuario mediante ‘id’, ‘get_user_tweets_by_user_id_from_db’ obtener los tweets de un usuario por el ‘id’ del mismo. El resto de los métodos se han usado para realizar las pruebas, pero no son relevantes en la aplicación.

- **Fase 3**: Obtención y procesado de atributos de la Base de Datos

  Esta es de las fases más importantes de la herramienta en la que se implementan los métodos que procesan los datos para obtener los atributos del dataset.

  El fichero ‘twitter_data.py’ tiene la clase TwitterData con los atributos y métodos correspondientes.

  El atributo de esta clase es ‘api’, que crea el objeto de la librería ‘Twython’ con los tokens y claves especificados en la fase 0 de configuración para acceder a la API de Twitter.

  Los métodos que se implementan son los siguientes:

  - ‘drop_data’: Borra las tablas ‘usuarios’ y ‘tweets’ de la Base de datos.
  - ‘get_user_and_tweets_dataset’: Este método devuelve el documento en formato DataFrame de la librería ‘pandas’ de los atributos pertenecientes al dataset.

    Primeramente, obtiene todos los usuarios de la colección de ‘usuarios’ de la Base de Datos. Para cada ususario, obtiene su información (llamando a ‘get_user_data’, explicada posteriormente) y sus tweets.

    Con estos datos crea un dataset en forma de diccionario con una lista por cada atributo de entrenamiento. En cada lista se almacena el atributo correspondiente de todos los usuarios.

    Finalmente, la función devuelve este dataset en forma de DataFrame.
  - ‘get_user_data’: Este método devuelve, pasándole el objeto usuario y timeline con los tweets, el documento en formato JSON de todos los atributos de clasificación de un usuario de Twitter. Calcula el ratio friends/followers, número de tweets publicados, número de favoritos marcados, días desde que se creó su cuenta en Twitter, listas en las que se encuentra el usuario y todos los atributos correspondientes a la información de sus tweets. Estos se procesan y

calculan en la función ‘analyze_timeline’ y son: media de hashtags en sus tweets, media de urls en sus tweets, frecuencia a la que tweetea, aplicación desde donde tweetea más frecuentemente, media de tiempos entre que se ha creado le tweet y el usuario lo ha retweeteado, ratio retweets/tweets, número medio de menciones a usuarios en sus tweets y numero de respuestas a usuarios.

- ‘analyze_timeline’: Las variables relacionadas con el timeline y, comentadas anteriormente, se calculan y devuelven en este método. Todas las medias calculadas se realizan sumando los datos que se obtienen del objeto tweet en un bucle por cada tweet en el timeline y posteriormente se dividen entre el total de tweets. Los ratios simplemente dividiendo los dos datos deseados. Para calcular el origen de los tweets, se introduce el origen de cada tweet en una colleccion y se llama al método ‘max_dictionary’ explicado posteriormente para obtener el origen de tweeteo más frecuente. La frecuencia media de tweeteo se obtiene de llamar a la función ‘frequence_average’ con la lista de timestamps de los tweets. De igual forma, el método ‘time_between_created_and_retweeted’ devuelve la media de tiempos entre la creación del tweet y el retweet del usuario. Se ayuda de la función ‘get_time_between_created_and_retweeted’ que resta los tiempos.
- ‘get_user_information’: Este método accede a la API de Twitter para obtener el objeto ‘usuario’ y su timeline con objetos ‘tweet’. Con ello se llama al método ‘get_user_data’ que devuelve el documento con los atributos calculados. La función devuelve este documento en forma de DataFrame y además, se introduce en el mismo el atributo ‘clase’ etiquetado como ‘NaN’ (sin valor) puesto que se trata del usuario que va a ser testeado y clasificado en la ejecución.
- ‘frequence_average’: A partir de una lista de timestamps devuelve, calculando en un bucle, la media de las diferencias entre uno y el posterior. Con ello se sabe la frecuencia a la que tweetea un usuario.
- ‘get_time_between_created_and_retweeted’: Calcula, a partir de dos fechas dadas por parámetro, la diferencia, en segundos, entre ellas. Se utiliza para calcular el tiempo que ha pasado desde que se ha creado un tweet y el usuario lo ha retweeteado.
- ‘time_between_created_and_retweeted’: A partir de una lista de timestamps, los sumey divide entre el total para calcular la media de tiempos entre la creación de un tweet y el retweet del usuario.
- ‘max_dictionary’: Devuelve, a partir de un diccionario, la clave y el valor que más se repiten en el mismo. Se usa para obtener el origen de tweeteo más frecuente del diccionario que se le pasa.
- ‘get_days_from_creation_date’: Calcula, a partir de una fecha dada por parámetro, los días que han pasado desde la misma hasta la fecha actual. Se utiliza para calcular los días de creación de la cuenta de Twitter.

- **Fase 4**: Implementación de métodos para la gestión de clasificadores.

Esta fase engloba la implementación de la funcionalidad relacionada con los clasificadores; generar los modelos, predecir los resultados, medir la precisión de un dataset y generar las gráficas de las matrices de confusión.

Los métodos del fichero 'classifiers.py' realizan estas funcionalidades con ayuda de 'classifier_models.py' para preservar y obtener los resultados que se requieren. Los cuatro métodos principales son los siguientes:

- 'prediction_clasifiers': Devuelve un diccionario con las predicciones de cada uno de los clasificadores utilizados. Por cada uno de ellos llama a su propio método 'prediction_*classifier_name*_classifier' que carga el correspondiente modelo generado previamente para ajustarlo con el dataset de entrenamiento ('fit') y posteriormente predecir ('predict') el resultado con el dataset de test de entrada.
- 'precision_clasifiers': Devuelve un diccionario con las precisiones de cada uno de los clasificadores utilizados. Por cada uno de ellos llama a su propio método 'precision_*classifier_name*_classifier' en el que carga el correspondiente modelo generado previamente para ajustarlo con el dataset de entrenamiento ('fit') y posteriormente medir la precisión ('score') con el dataset de test de entrada y salida.
- 'generate_models_classifiers': Guarda en la carpeta 'Modelos' los modelos generados por cada uno de los clasificadores. Este método llama a todos ellos de la forma 'save_model_*classifier_name*_classifier' que obtiene el modelo correspondiente al clasificador gracias a la librería scikit-learn, lo entrena con el dataset de entrenamiento ('fit') y llama al método 'save_models' del módulo 'classifier_models' pasándole el modelo generado y entrenado, y el nombre del clasificador.
- 'precision_confusion_matrix_classifiers': La funcionalidad de este método es generar de forma gráfica las matrices de confusión de cada clasificador para tener una referencia visual de los resultados positivos y negativos. Primero obtiene las predicciones llamando al método 'prediction_clasifiers' con el dataset de entrenamiento y el dataset de entrada de test para obtener las predicciones. Posteriormente por cada uno de los clasificadores llama al método 'get_confusion_matrix' pasándole el dataset de test de salida y la predicción del clasificador para comparar ambas y obtener los resultados correctos y erróneos. Dicho método genera las gráficas de la matriz de confusión sin normalizar y normalizada correspondientes en formato PNG y guardándolas en la carpeta 'Graphics'.

Otros métodos auxiliares de esta fase son:

- 'encode_attributes': Este método recibe el dataset de entrenamiento y el de test. Su función es codificar los atributos que no son numéricos, como en nuestro caso es 'frecuence_source'. Gracias a la herramienta de scikit-learn 'preprocessing.LabelEncoder' podemos numerar estos atributos primero ajustando los valores del atributo ('fit') y posteriormente transformándolos ('transform') a numérico.
- 'split_train_test': Este método divide el dataset en dos partes perteneciendo una parte a 'entrenamiento' y otra a 'test'. Se utiliza para testear el dataset de la

base de datos. Así mismo recibe la proporción (comprendida entre 0 y 1) que quiere ser dividida en el dataset de entrenamiento. Devuelve ambos datasets por separado.

- 'read_dataset': Llama al método 'load_dataset' del módulo 'classifier_models' para obtener el dataset guardado dado el nombre de este.
- 'preserve_dataset': Llama al método 'save_dataset' del módulo 'classifier_models' para guardar el dataset pasándole el nombre deseado.

Cabe destacar que el fichero 'classifier_models.py' tiene los métodos para guardar y cargar los modelos en las carpetas correspondientes. El método 'save_dataset' guarda el dataset tanto con el nombre de 'dataset*_classifier_name*' como con el nombre 'dataset'. El primero se utiliza para testear el dataset diferenciando los clasificadores, y el segundo es cargado para testear el usuario. Para esto, siempre se va a cargar el fichero con nombre 'dataset' que corresponderá a la última ejecución de 'generate_models' del módulo 'test'.

- **Fase 5**: Módulo principal 'Testeo' con la gestión de llamadas al resto de módulos.

Esta parte corresponde a la integración del resto de módulos en el fichero 'test.py' con tres métodos; uno para cada funcionalidad. Estos son:

- 'generate_models': Este método contiene la funcionalidad de llamadas al resto de módulos para generar y guardar los modelos de los clasificadores. Se le pasa el nombre de la base de datos a la que se quiere acceder.

  Primero se obtiene el dataset de entrenamiento llamando al método 'get_user_and_tweets_dataset' del módulo 'TwitterData' pasándole el nombre de la base de datos. Este dataset lo guarda para su posterior uso con el método 'preserve_dataset' del módulo 'classifiers'. Posteriormente codifica los atributos no numéricos, que son 'frecuence_source' y la clase del dataset 'class' ('bot' o 'human') y separa el dataset en dataset de entrenamiento de entrada y dataset de entrenamiento de salida para finalmente llamar a 'generate_models_classifiers' que realiza el resto de la funcionalidad guardando los modelos.

- 'test_user': Este método testea el usuario cuyo nombre es pasado por parámetro.

  Accede tanto al módulo 'classifiers' con el método 'read_dataset('dataset')' para obtener el dataset de entrenamiento guardado en la ejecución del método 'generate_models' como a 'TwitterData' para obtener la información del usuario pasado por parámetro (accediendo a la API de Twitter) que será el dataset de test. Posteriormente codifica los atributos no numéricos y se guardan las entradas del dataset de test. Con los dos datasets se llama al método 'prediction_clasifiers' del módulo 'classifiers' para obtener las predicciones de salida, que son devueltas.

- 'test_dataset': Este método testea el dataset cuyo nombre es pasado por parámetro. Primeramente, llama al método 'read_dataset' del módulo 'classifiers' con el nombre del dataset para obtenerlo y después al método 'split_train_test' junto con una proporción pasada por parámetro para dividir el dataset en: entrenamiento y test. Posteriormente se codifican los atributos no

numéricos con ‘encode_attributes’ y, separando las entradas y salidas de ambos datasets, se llama a los métodos ‘precision_clasifiers’ y ‘precision_confusion_matrix_classifiers’ para obtener las precisiones, que serán devueltas, y las matrices de confusión generadas, que serán guardadas, respectivamente.

- **Fase 6**: Creación de la pequeña aplicación web; controladores y vistas.

Una vez realizada la funcionalidad de la herramienta, se ha realizado una pequeña interfaz con ayuda del framework Flask para la estructura y Bootstrap para el diseño y así poder visualizar los resultados.

Siguiendo el patrón MVC (Modelo-Vista-Controlador), se ha creado un fichero ‘controllers.py’ con los controladores necesarios y la carpeta ‘templates’ con las vistas ‘html’ para renderizar el resultado de los controladores.

Los controladores acceden a los métodos del módulo ‘test’ teniendo un controlador para cada uno de ellos. Son los siguientes:

- ‘index’: Con la ruta raíz, ‘/’, simplemente renderiza la vista ‘index.html’ con el formulario inicial.
- ‘testuser’: Con la ruta ‘/testuser’, y utilizado el método ‘POST’ obtiene del formulario el nombre del usuario introducido para llamar al método ‘test_user’ del módulo ‘test’. Posteriormente renderiza los resultados en la vista ‘testuser.html’, que son las predicciones de los clasificadores.
- ‘generatemodels’: Con la ruta ‘/generatemodels, y utilizado el método ‘POST’ obtiene del formulario el nombre de la base de datos para llamar al método ‘generate_models’ del módulo ‘test’. Posteriormente renderiza los resultados en la vista ‘generatemodels.html’, que es un mensaje informando de que se han generado los modelos.
- ‘testdataset’: Con la ruta ‘/testdataset, y utilizado el método ‘POST’ obtiene del formulario el nombre del dataset que se quiere testear así como la proporción para dividirlo. Posteriormente llama al método ‘test_dataset’ del módulo ‘test’. Finalmente renderiza los resultados en la vista ‘testdataset.html’, que son las precisiones de los clasificadores para ese dataset y proporción.

La aplicación se lanza desde el fichero ‘runserver.py’ en local, puerto 8080.

Los resultados del renderizado de los controladores se muestran en el diseño de la interfaz, apartado 3.5.

5 Integración, pruebas y resultados

Para probar el funcionamiento de la herramienta se ha realizado una batería de pruebas a medida que se va realizando el desarrollo. Primero unitaria, después de la implementación de cada módulo y, finalmente de integración con todos los módulos.

5.1 Pruebas unitarias

Como pruebas unitarias, se han testeado los módulos importantes uno por uno, tras su realización.

En primer lugar, la precarga en la Base de Datos a partir de un fichero dio problemas puesto que había usuarios no existentes o sin suficientes tweets para ser clasificados. Se realizó un método 'parse_file' para limpiar ese fichero inicial y ya posteriormente, ser cargados en la Base de Datos.

Esta carga dio como resultado dos colecciones con un gran número de registros para ser probados, por lo que se optó por volcar parte de esos datos en otras dos Bases de Datos: 'prueba' y 'twitter'. La Base de Datos principal sobre la que finalmente se ejecuta la herramienta es 'bigtwitter', con un total de 2238 usuarios y 395142 tweets de esos usuarios procedentes de Twitter. Para la visualización de los datos en la Base de Datos, se ha usado 'robo3t'.

Antes que nada, se han probado las conexiones a la Base de Datos (local) y la API de Twitter con los tokens generados. Una vez que han sido configurados correctamente, se procede a probar los módulos.

Se realizó una batería de pruebas del módulo 'Librería Base de Datos' probando todos los métodos para comprobar los resultados que devuelve la Base de Datos.

Las pruebas del módulo 'Cálculo de atributos' se realizan primero contra la Base de Datos 'prueba' puesto que implementan la lógica para crear el dataset a partir de los objetos 'users' y 'tweets' y no es necesario comprobar su funcionamiento con una Base de Datos grande.

De igual manera, los métodos del módulo 'Clasificadores' que obtienen los modelos y miden las precisiones y predicciones han sido probados en una batería aparte con cada uno de los clasificadores.

Se ha creado el fichero 'TwitterException.py' con varias excepciones que controlan una ejecución errónea en la herramienta. Son: 'UserAccountProtected' cuando la cuenta del usuario, al acceder a la API está protegida, 'UserNotExists' cuando el usuario no existe, 'DatabaseNotExists' cuando la Base de Datos no existe o no está bien configurada, 'DatasetNotExists' cuando un dataset no existe o 'LimitPercentage' cuando se ha

introducido una proporción errónea al testear el dataset. La implementación de estas excepciones ha permitido controlar y debuggear mejor el desarrollo del trabajo.

5.2 Pruebas de integración

La integración del resto de módulos se ha ido realizando a medida que uno de ellos necesitaba usar métodos de otro, entonces, se probaban conjuntamente. La integración final se ha realizado en el módulo 'Testeo' quien es el que, en sus métodos, devuelve los resultados necesarios y objetivo de la herramienta. Las pruebas de integración ya se realizan contra la Base de Datos 'bigtwitter'.

Con ayuda de la interfaz lanzada en el servidor local, una vez completado todo el desarrollo, se ha implementado la funcionalidad para generar matrices de confusión y poder ver los resultados.

5.3 Resultados

Se han tomado varias ejecuciones contra el dataset correspondiente a la Base de Datos 'bigtwitter' para testear el dataset y ver en su matriz de confusión los resultados con varios valores de entrada:

- Con proporción 0.5 en la figura 16:

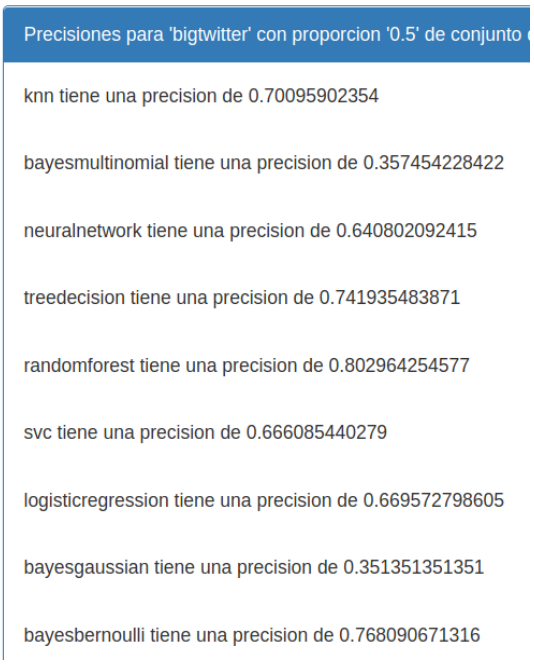

Figura 16. Precisión del dataset con proporción 0.5.

- Con proporción 0.6 en la figura 17:

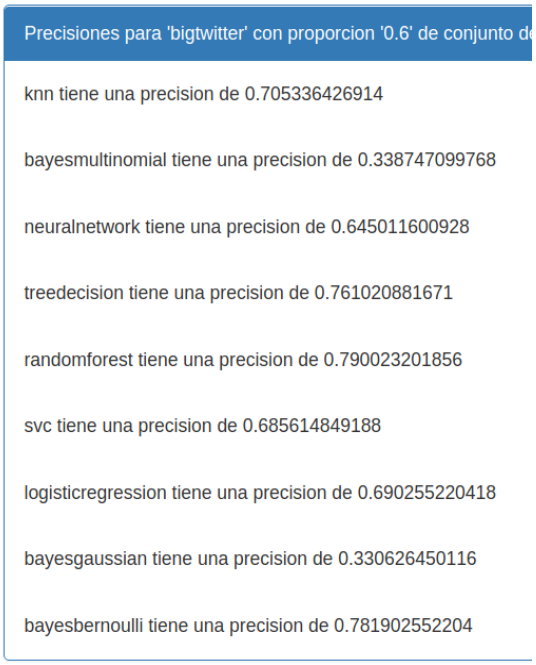

Figura 17. Precisión del dataset con proporción 0.6.

- Con proporción 0.7 en la figura 18:

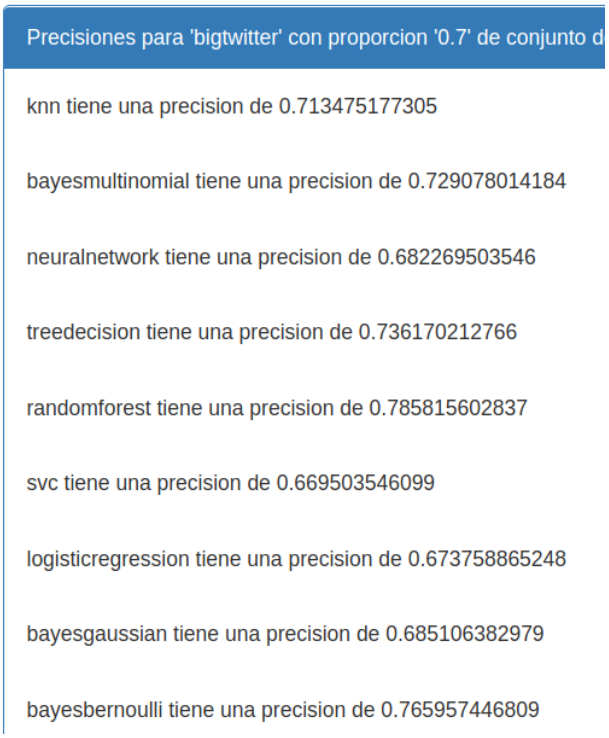

Figura 18. Precisión del dataset con proporción 0.7.

- Con proporción 0.8 en la figura 19:

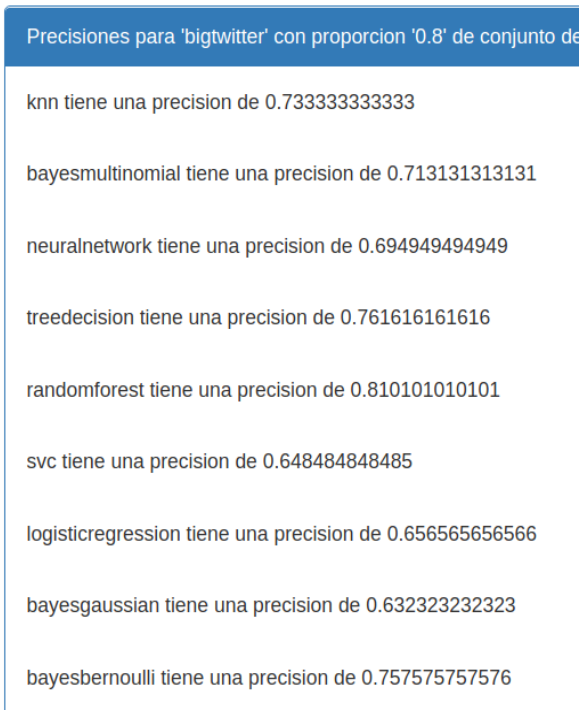

Figura 19. Precisión del dataset con proporción 0.8.

- Con proporción 0.9 en la figura 20:

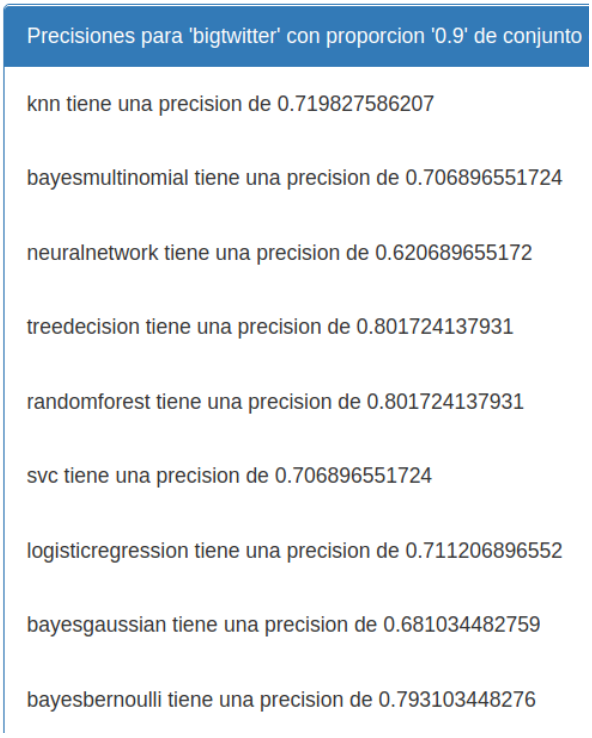

Figura 20. Precisión del dataset con proporción 0.9.

Observamos que el clasificador 'Random Forest' obtiene los mejores resultados. Veamos las matrices de confusión:

- Con proporción 0.6:

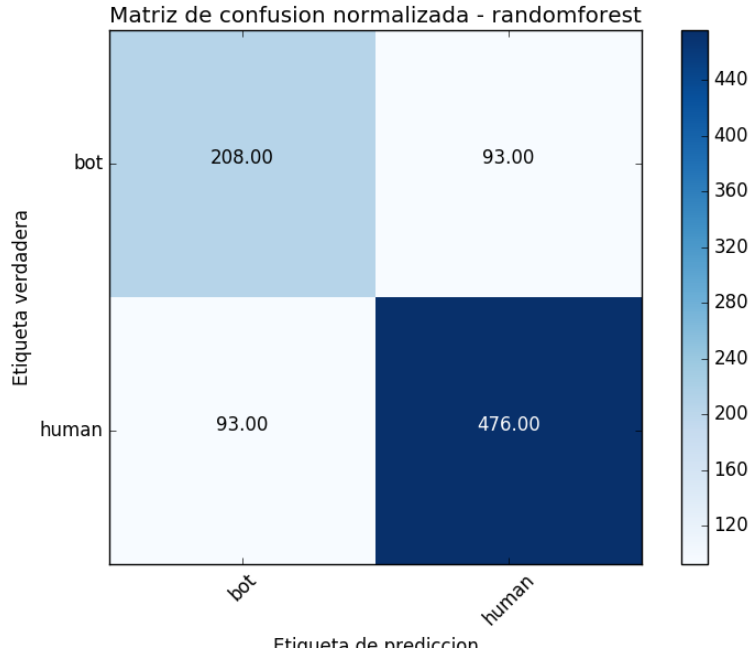

Figura 21. Matriz de confusión para Random Forest con proporción de entrenamiento 0.6.

- Con proporción 0.85:

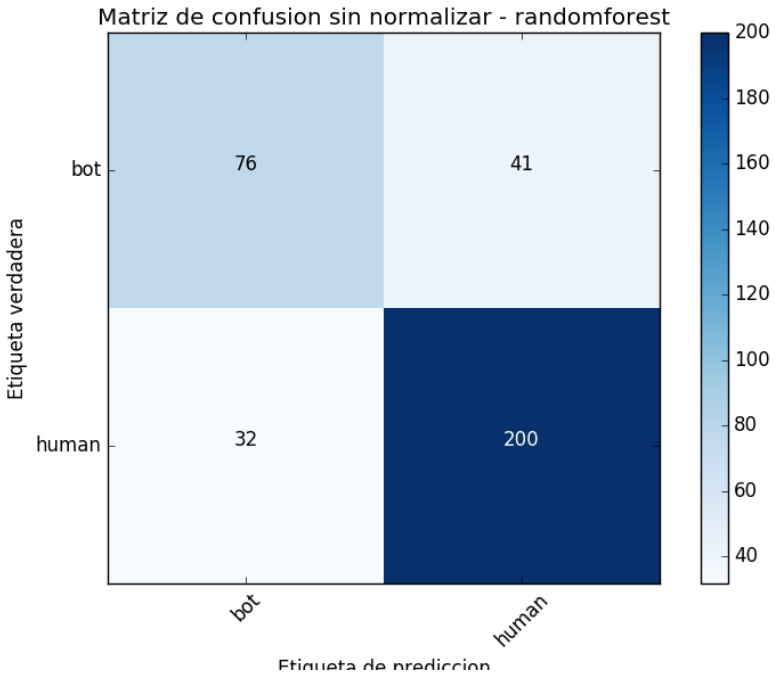

Figura 22. Matriz de confusión para Random Forest con proporción de entrenamiento 0.85.

Los resultados en la predicción de ‘bot’ son bastante buenos. En la figura 21 se obtienen 208 verdaderos positivos sobre 93 falsos positivos. En la figura 22 los resultados son de 76 verdaderos sobra 41 falsos positivos. Todavía es mejor los resultados para ‘human’ siendo de 476 verdaderos sobre 93 falsos en la figura 21 y 200 verdaderos sobre 32 falsos en la figura 22. Los resultados obtenidos son bastante buenos y ‘Random forest’ parece el modelo adecuado para fijarse a la hora de predecir un usuario nuevo. El estudio de Ulf Aslak Jensen y Christopher Frederick Wiin Schenk en [24] coincide en la elección de Random Forest.

Vemos ahora los atributos más importantes en la precisión de ‘Random Forest’. Las figuras 23 y 24 muestran resultados para proporcion de 0.7 y 0.9:

	importance
favourites_count	0.214826
ratio_retweets_to_tweets	0.184664
hashtags_average	0.152965
time_between_created_and_retweeted	0.111322
reply_count	0.101470
urls_average	0.083835
user_mentions	0.076617
ratio_friends_to_followers	0.031699
frecuence_source	0.018472
frecuence_tweeting	0.009870
listed_count	0.006344
statuses_count	0.005086
days_from_created	0.002831

Figura 23. Atributos más importantes para Random Forest con proporción 0.7 en entrenamiento.

	importance
favourites_count	0.371237
urls_average	0.291541
ratio_retweets_to_tweets	0.113030
hashtags_average	0.072546
reply_count	0.067763
listed_count	0.031394
ratio_friends_to_followers	0.022965
user_mentions	0.017209
frecuence_source	0.006376
days_from_created	0.002642
frecuence_tweeting	0.002163
statuses_count	0.001134
time_between_created_and_retweeted	0.000000

Figura 24. Atributos más importantes para Random Forest con proporción 0.9 en entrenamiento.

Vemos que el numero de favoritos, la media de url's y hashtags por tweet y el ratio retweets/tweets son las características más importantes en esta ejecución.

Estos resultados parecen confirmar las suposiciones del estudio presentado en la sección 2.1.4.2 respecto a los atributos relevantes para identificar bots.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo se ha visto una manera de intentar detectar cuentas automatizadas en la red social Twitter. Se han estudiado los posibles patrones que involucran los seguidores falsos en esta comunidad y comprobamos que las suposiciones hechas en el apartado 2.1.4.2 corresponden en cierta medida con los resultados obtenidos en el apartado 5.3 y más concretamente visualizándose en las figuras 23 y 24.

Finalmente se han construido varios modelos de clasificación para identificar dichos patrones en nuevos usuarios.

La tarea de analizar los atributos es clave en todo el diseño y desarrollo y, en especial, lo es en el ámbito del 'Machine learning'. Existen herramientas tecnológicas muy potentes para trabajar en este sector, sin embargo, de momento no está muy desarrollado y puede llegar a ser el centro de la tecnología en un futuro que quizá no esté tan lejos como pensamos.

El gran problema de ello es el manejo de grandes cantidades de datos. En el trabajo se ha tenido que persistir la información de alguna manera puesto que ralentiza mucho la ejecución de la herramienta si se realiza el proceso de generación de modelos desde el principio. Ahora se ha trabajado sobre un conjunto de datos infinitamente más pequeño del que puede llegar a ser tratado con tecnologías adecuadas. Aquí entra el concepto de 'Big Data'. Es muy importante, cuando se tienen grandes cantidades de datos, un procesado y búsqueda óptimos en ellos.

La realización de este proyecto me ha ayudado a entender más profundamente estos dos conceptos que están por desarrollar. He trabajado con nuevas herramientas que han permitido aprender técnicas y descubrir nuevas ideas sobre las que podrían ser aplicadas.

6.2 Trabajo futuro

El potencial de la herramienta de detección de usuarios falsos es muy grande y se puede llegar a obtener mejores resultados estudiando, como he comentado antes, patrones nuevos que identifiquen a estos 'bots'.

Tener una Base de Datos óptima es imprescindible en la herramienta ya que los modelos se construyen en función a esta. Un objetivo futuro es conseguir una base de datos actualizada y de mayor tamaño con la que poder trabajar mejor.

No se ha profundizado en exceso en el estudio de los clasificadores. Es imprescindible conocerlos para saber cuál utilizar en las circunstancias requeridas.

Me gustaría poder mejorar la aplicación consiguiendo esos objetivos y, además, se puede incluir e integrar otro tipo de herramientas. Por ejemplo, el estudio del sentimiento de los tweets o de las 'fake news' en la red social.

Referencias

[1] Classmates, www.classmates.com

[2] Sixdegrees, www.sixdegrees.com

[3] LinkedIn, www.linkedin.com

[4] Facebook, www.facebook.com

[5] Twitter, www.twitter.com

[6] Instagram, www.instagram.com

[7] Snapchat, www.snapchat.com

[8] The Social Media Family, "IV Estudio sobre los usuarios de Facebook, Twitter e Instagram en España", Febrero de 2018.

[9] Statisa, "Panorama mundial de las redes sociales", febrero de 2018

[10] Fernando Díaz Gomez, "Aprendizaje auromático: métodos y aplicaciones".

[11] Raúl E. López Briega, "Machine Learning con Python".

[12] Twython, https://twython.readthedocs.io

[13] Erin Shellman, "Bot or not, an end-to-end data analisys in Python", 17 de Agosto de 2015.

[14] Xakata, "A la caza de bots en Twitter, pistas, métodos y por qué es tan difícil detectarlos todos.", 8 de Noviembre de 2016.

[15] Aalto University, Zafar Gilani, "Measuring, characterising and detecting automated agents on Online Social Networks"

[16] scikit-learn, http://scikit-learn.org/stable/index.html

[17] Flask, http://flask.pocoo.org/

[18] MongoDB, https://www.mongodb.com/

[19] User Object, API Twitter, https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object

[20] Tweet Object, API Twitter, https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html

[21] Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, Alessandro Flammini, "Online Human-Bot Interactions: Detection, Estimation, and Characterization", 9 marzo 2017.

[22] Chao Michael Zhang, Vern Paxson, "Detecting and Analyzing Automated Activity on Twitter", 20 Marzo 2011.

[23] Zafar Gilani, Ekaterina Kochmar, Jon Crowcroft, "Classification of Twitter Accounts into Automated Agents and Human Users", 13 Julio 2017.

[24] Ulf Aslak Jensen, Christopher Frederick Wiin Schenk, "Random Forest classification of Twitter users to detect features linked to bot susceptibility".

[25] Botometer, https://botometer.iuni.iu.edu

[26] Pandas, https://pandas.pydata.org/

[27] NLTK, http://www.nltk.org/

[28] Botcheck, https://botcheck.me/

[29] Ash Bhat, Rohan Phadte, “An Analysis of Propaganda Bots on Twitter”.

Glosario

API	Application Programming Interface
BD	Base de Datos
IA	Inteligencia Artificial
Dataset	Conjunto de datos
MVC	Modelo-Vista-Controlador
CSV	Comma-separated Values, Valores separados por coma

Anexos

A Manual de configuración

En el fichero ‘config.py’ se deben establecer:

- Los tokens y claves generados por tu aplicación de Twitter.
- Usuario, contraseña, host y puerto de la base de datos para MongoDB.
- Host y puerto para lanzar la aplicación Flask.

Para lanzar la aplicación, desde el directorio base, en la terminal de comandos, ejecutar: ‘$> python runserver.py’. Se abrirá una conexión en el host y puerto indicados.

Si todo ha ido bien, deberá abrirse la pantalla índice con el formulario inicial y podrás utilizarlo.